

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA
EL ESCANEADO DE DOCUMENTOS MEDIANTE UN
CLIENTE ANDROID**

1415_098_COSI

Aristeo Mateos Rodríguez

Tutor: David Arroyo Guardado

Mayo 2015

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA EL ESCANEADO DE DOCUMENTOS MEDIANTE UN CLIENTE ANDROID

Aristeo Mateos Rodríguez

Tutor: David Arroyo Guardeno

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo 2015

Resumen

Resulta asombroso a la par que preocupante el darse cuenta de lo necesario e imprescindible que es para la sociedad actual el uso de la avanzada tecnología de la que dispone. El mundo se ha acomodado al empleo de una tecnología que evoluciona a cada segundo que pasa y que permite, desde realizar múltiples copias de un documento de forma casi instantánea, hasta ayudar a identificar mejor posibles tratamientos de enfermedades o a mejorar los ya existentes.

No es de extrañar que compañías de todas las ramas tecnológicas aprovechen esta necesidad de la sociedad, y busquen desarrollar todo tipo de herramientas o aplicaciones que realizan o le facilitan la realización de tareas al ser humano, un claro ejemplo son las herramientas de digitalización de contenidos, que permiten automatizar el procesado de información, lo que implica mayor velocidad a la hora de leer textos y clasificar la información.

Uno de los sectores tecnológicos que más crecimiento está experimentando en la actualidad es el de desarrollo de aplicaciones para dispositivos móviles, que permiten unificar diferentes herramientas en un solo aparato que cabe en la palma de tu mano. Los usuarios cada vez son más conscientes de la variedad de herramientas que tienen a su disposición, lo que les hace más exigentes a la hora de decantarse por unas u otras y de esta forma atender a factores como la usabilidad, la curva de aprendizaje de la herramienta o los medios necesarios para usarla, que en conjunto dan lugar al grado de *perfección tecnológica*.

El objetivo que se persigue en este proyecto es, en primer lugar, diseñar y desarrollar una aplicación que permita al usuario el escaneado, almacenamiento y compartición de documentos empleando como único medio su *Smartphone*. En segundo lugar se pretende alcanzar un alto grado de perfección tecnológica, simplificando la interacción del usuario con la herramienta que puede reducirse a saber usar la cámara del móvil, tanto para la autenticación mediante códigos personalizados, como para el escaneado de documentos, permitiendo lograr un aprovechamiento del potencial del *Smartphone*.

El siguiente documento recoge la descripción en detalle del proyecto, incluyendo la explicación de las actividades realizadas y las fases seguidas para desarrollo de la aplicación, desde un análisis inicial hasta las pruebas realizadas sobre el proyecto y conclusiones finales.

Palabras clave: Dispositivos móviles, Android, aplicación, herramienta, escáner, autenticación, códigos QR, códigos de barras, usable, e-learning.

Abstract

Nowadays technology is of major relevance in our society, which represents both a whole set of opportunities and a risk. People use technology in everyday activities and technology is changing and improving day by day. Certainly, today it is possible to accomplish in much faster way activities as document treatment and copy, but also technology provides a means to better address complex problem as diseases study.

It is not surprising that technological companies exploit this society need, and try to develop all kinds of tools and applications that make it easy to humans to perform some tasks. As a clear example of such a situation we have the tools for content digitalization, which enable the automatic processing of information and make more efficient and accurate text interpretation and information classification.

Currently, the development of applications for mobile devices is one of the fastest-growing technology sectors. Mobile devices supply a means to integrate different tools in one device that fits in the palm of your hand. Users are getting aware of the variety of tools at their disposal, making them more demanding when opting for one or another tool and thus they take in account factors such as usability, the learning curve of the tool, or the hardware needed to use it. That is to say, users are concerned with the technological perfection of the tools.

The objective pursued in this project is, first of all, design and develop an application that allows the user to scan, store and share documents using their *Smartphone*. Secondly along the work we want to achieve a high degree of technological perfection, which implies the simplification of user interaction with the tool. This goal demands an initial study of potential users of the tool. This initial study is finally apply to design a software tool to help users to use the camera for login by means of custom code, and for scanning documents.

The following document provides a detailed description of the project including an explanation of the activities and the steps followed in the application development, from initial analysis to the project testing and final conclusions.

Keywords: Mobile devices, Android, application, tool, Scanner, authentication, QR codes, barcodes, usable, e-learning.

Agradecimientos

Tras concluir mi trabajo de fin de grado, quiero agradecer a los culpables de que de un modo u otro éste haya sido posible.

A David Arroyo, por ayudarme y tirar de mí cuando el desarrollo del proyecto se ponía cuesta arriba.

A WWW por permitirme vivir un sueño y por darme fuerza semana tras semana.

A mis padres y hermana por apoyarme y confiar en mí, por esforzarse para darme la oportunidad de vivir un viaje y una experiencia que nunca olvidaré y por hacer de mí la persona que soy hoy en día.

Y a Beatriz, mi apoyo 24/7.

A todos, Gracias.

Glosario

<u>Acoplamiento</u>	Es el grado de interdependencia que se da entre las interfaces en un desarrollo software. Su disminución potencia la reutilización de código y reduce la dependencia entre las distintas interfaces de programación.
<u>API</u>	<i>Application Programming Interface</i> . Conjunto de subrutinas, funciones y métodos que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción.
<u>APK</u>	Paquete Android (Android Package).
<u>App</u>	Aplicación.
<u>Array</u>	Tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo.
<u>Autenticación</u>	Proceso de verificación de la identidad digital del remitente de una comunicación mediante una petición para conectarse.
<u>Código QR</u>	Código de rápida respuesta (Quick Response).
<u>Cohesión</u>	Es el grado de agrupamiento de unidades del software en otras unidades mayores.
<u>Document imaging</u>	Tecnología de la información capaz de replicar documentos y convertirlos a imágenes digitales.
<u>Escuchador</u>	Método de los elementos de la interfaz que se ejecutan cuando éstos son accionados.

<u>FTP</u>	<i>File Transfer Protocol</i> . Protocolo de transferencia de ficheros por una red basada en TCP. (<i>Transmission Control Protocol</i>)
<u>Hosting</u>	Servicio que provee a los usuarios de Internet un sistema para poder almacenar cualquier contenido accesible vía web.
<u>MVC</u>	Modelo Vista Controlador
<u>Parsear</u>	Analizar una secuencia de símbolos a fin de determinar su estructura gramatical con respecto a una gramática formal dada.
<u>SDK</u>	Kit de desarrollo software (<i>Software Development Kit</i>).
<u>SFTP</u>	Protocolo seguro de transferencia de ficheros debido a su empaquetado con SSH. (<i>SSH File Transfer Protocol</i>)
<u>Smartphone</u>	Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos, de realizar actividades y una mayor conectividad que un teléfono móvil convencional.
<u>SSL</u>	Protocolo criptográfico que proporciona comunicaciones seguras por la red (<i>Secure Sockets Layer</i>).
<u>Usuario (Android)</u>	Individuo que hace uso del sistema operativo Android.

Índice general

Resumen	V
Abstract	VII
Agradecimientos	IX
Glosario	XI
Índice general	XIII
Índice de figuras	XV
Índice de tablas	XVI
1. Introducción	1
1.1 Motivación	1
1.2 Objetivos	3
1.3 Planificación del proyecto	4
1.4 Estructura de la memoria	5
2. Estado del arte	7
2.1 Android	7
2.2 Digitalización de documentos	9
2.3 Almacenamiento de datos en servidor	10
2.4 Compartición de documentos	12
2.5 Códigos digitales para la identificación de recursos	13
2.6 Mercado actual	13
3. Desarrollo del sistema	15
3.1 Análisis	15
3.1.1 Análisis de usuarios	15
3.1.2 Análisis de requisitos	16
3.1.2.1 Requisitos funcionales	17

3.1.2.2	Requisitos no funcionales	20
3.1.3	Análisis de herramientas y tecnologías	22
3.2	Diseño	23
3.2.1	Diseño de la arquitectura del sistema	23
3.2.2	Diseño del modelo de datos del sistema	25
3.2.2.1	Elementos del sistema	25
3.2.2.2	Estructura de datos del sistema	26
3.2.3	Diseño de la interfaz del sistema	27
3.2.4	Diseño de la lógica del sistema	29
3.3	Implementación	30
3.3.1	Preparación del entorno	31
3.3.2	Servidor	34
3.3.2.1	Funciones de gestión	35
3.3.2.2	Funciones de administración de archivos	36
3.3.2.3	Funciones de administración de grupos	38
3.3.3	MVC	39
3.3.3.1	Implementar el modelo de la aplicación	39
3.3.3.2	Actualizar el fichero de manifiesto	40
3.3.3.3	Implementar la vista de la actividad	41
3.3.3.4	Implementar el controlador de la actividad	44
3.3.3.5	Codificaciones a destacar	44
3.3.4	Incrementos de software utilizable	46
4.	Evaluación del sistema	47
4.1	Plan de pruebas	47
4.1.1	Pruebas unitarias	47
4.1.2	Pruebas de integración	48
4.1.3	Pruebas de sistema	48
4.1.4	Pruebas de validación	49
4.2	Prueba de los incrementos de software utilizables	49
4.2.1	Incremento 1	49
4.2.2	Incremento 2	50
4.2.3	Incremento 3	50
4.2.4	Incremento 4 (Aplicación final)	51
5.	Conclusiones	53
5.1	Valoración personal del proyecto	53
5.2	Conocimientos y valores aportados	53
5.3	Futuras versiones	54
	Bibliografía	55
	Anexo A. Diagrama de Gantt	57

Índice de figuras

Figura 1. Evolución del número de aplicaciones disponibles. Fuente: readwrite.com. ...	1
Figura 2. Visualización del potencial del <i>Smartphone</i> para tareas como el escaneado y almacenamiento. Fuente: Propia.	2
Figura 3. Marco de desarrollo ágil empleado (4 sprints). Fuente: Propia.	4
Figura 4. Ventas mundiales de Smartphones. Fuente:Wikipedia.....	8
Figura 5. Versiones de Android y su porcentaje de distribución. Fuente: AndroidStudio.	9
Figura 6. Pantelégrafo <i>Fuente: Flickr</i>	9
Figura 7. File sharing empleado. Fuente: Propia.....	12
Figura 8. Código QR. Fuente: Propia.....	13
Figura 9. Patrón de arquitectura de software de la aplicación. Fuente: Propia.	24
Figura 10. Diagrama de clases de la aplicación. Fuente: Propia.	26
Figura 11. Diagrama ER. Fuente: Propia.	27
Figura 12. Mapa de navegación de la aplicación. Fuente: Propia.	28
Figura 13. Diseño de la interfaz de <i>login</i> . Fuente: Propia.	29
Figura 14. Simulación de interacción del controlador con el modelo y la vista. Fuente: Propia.....	30
Figura 15. Preparación del entorno (I) Fuente: AndroidStudio.....	31
Figura 16. Preparación del entorno (II). Fuente: AndroidStudio.	31
Figura 17. Preparación del entorno (III). Fuente: AndroidStudio.	32
Figura 18. Preparación del entorno (IV). Fuente: AndroidStudio.....	32
Figura 19. Preparación del entorno (V). Fuente: AndroidStudio.	33
Figura 20. Base de datos valija_virtual. Fuente: phpMyAdmin.....	34
Figura 21. Herramienta de renderizado para la implementación de la GUI. Fuente: AndroidStudio.	42
Figura 22. Pre visualización de tamaños de pantalla para portabilidad. Fuente: AndroidStudio.	42

Índice de tablas

Tabla 1. Variación de ventas trimestrales de dispositivos móviles <i>smartphone</i>	8
Tabla 2. Encuesta a los usuarios regulares	16

1. Introducción

1.1 Motivación

En la actualidad, se concibe a los *Smartphones* como una medio indispensable para el acceso a toda clase de información así como una herramienta que mediante sus aplicaciones, puede resultar útil en cualquier situación a la que puede enfrentarse una persona en su día a día.

Ninguna tecnología de consumo ha evolucionado tan rápidamente como el *smartphone* y Android ha estado en el centro de dicha evolución [1]. Cuando en 2008 se inició la era Android con el lanzamiento del primer dispositivo que lo integraba, el T-Mobile G1 [2], el sistema operativo prescindía de características que hoy en día son indispensables, como el teclado en la pantalla o la capacidad multitáctil. No obstante ya mostró algunas de sus características clave como la barra desplegable de notificaciones, los *widgets* en la pantalla de inicio o la integración con Gmail que proporcionaba una experiencia única hasta el momento con un cliente de correo. El éxito cosechado tras el lanzamiento del primer *smartphone* Android, permitió invertir recursos para mejorar progresivamente este nuevo sistema operativo, dando lugar a una evolución que continua hoy en día.

Con la llegada del *Android Market* y del *Apple App Store*, se inició la **revolución de las aplicaciones móvil** pues, hasta ese momento, no existía ninguna tienda centralizada que recogiese las aplicaciones disponibles para el dispositivo y desde entonces la cantidad de *apps* disponibles ha crecido exponencialmente cada año (Figura 1).

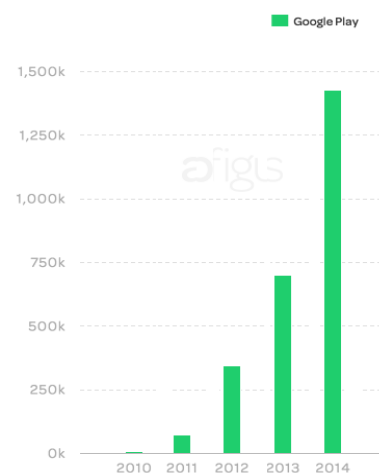


Figura 1. Evolución del número de aplicaciones disponibles. Fuente: readwrite.com.

No obstante, pese a que cada vez son más las personas que tienen acceso a un *smartphone*, se percibe **un uso limitado** de éstos por múltiples motivos. Ya sea por los usuarios que no ven en los dispositivos más que una forma de entretenimiento mediante juegos, o por usuarios para los que interactuar con las aplicaciones resulta una labor tediosa o incluso personas que se escudan en las barreras tecnológicas existentes para no confiar sus tareas diarias en la eficiencia de su móvil. En todos estos casos el resultado siempre es el mismo, el **desaprovechamiento del potencial del *smartphone***.

Por otra parte, se puede decir que vivimos en plena era de la información. Miles de personas **redactan, almacenan y comparten documentos día a día**, así pues el tratamiento de información hoy en día es fundamentalmente en formato digital. No obstante existen muchos entornos en los que el soporte de información es físico y, consecuentemente, se requiere su digitalización para su tratamiento con los medios TIC actuales, pero en muchas ocasiones esto no es factible por **no contar con mecanismos simples para la digitalización**. Los smartphones proporcionan esta forma simple de digitalizar información .

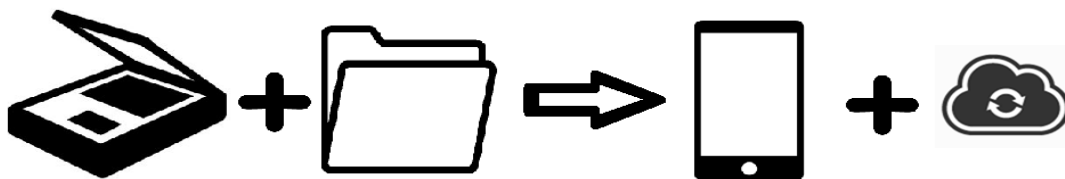


Figura 2. Visualización del potencial del *Smartphone* para tareas como el escaneado y almacenamiento. Fuente: Propia.

El proyecto vinculado a este Trabajo de Fin de Grado tiene por eje central la resolución de los problemas expuestos previamente mediante el **desarrollo de una aplicación Android** que aprovecha el potencial de los dispositivos a los que está orientada dicha aplicación y que **permite el escaneado de documentos** para su **almacenamiento** y posterior **compartición** (Figura 2).

Entre los motivos que justifican la realización de este proyecto se encuentran:

- **Existencia de una necesidad real:** que se verá resuelta con esta aplicación donde los usuarios podrán escanear los documentos desde la comodidad de su *Smartphone* y almacenarlos en cuestión de segundos.
- **Desconfianza o desconocimiento del potencial de un *Smartphone*:** ofrecer una interacción simple y atractiva que demuestre la utilidad del dispositivo y sus

eficaces resultados será una forma de acercar al usuario al uso de este tipo de aplicaciones.

- **Inexistencia de este tipo de aplicación:** que combina el escaneado de documentos con su almacenamiento y compartición.
- **Interés por el uso de tecnologías clave:** que emplearemos en el proyecto, como son el desarrollo web, la configuración de sistemas o el almacenamiento de datos en servidor, que son desconocidas para mí.
- **Interés personal por el desarrollo de aplicaciones Android:** que tanta utilidad presenta actualmente en plena evolución de dicho sistema operativo.
- **Interés de aprendizaje:** de todos los conocimientos que me puede aportar el desarrollo de este proyecto.

1.2 Objetivos

El objetivo de este trabajo es diseñar e implementar un sistema que permita el escaneado de documentos mediante el uso de un dispositivo móvil dotado de un sistema operativo Android. Paralelamente, se pretende hacer uso de herramientas que mejoren y simplifiquen la experiencia del usuario. Con ello se persigue demostrar el potencial de este tipo de tecnología, lo que se hará tratando de facilitar al usuario el empleo de aplicaciones móvil como medio para resolver situaciones de su día a día. Finalmente se propone una arquitectura para su uso en diversos ámbitos, como el académico.

Para la consecución de los objetivos propuestos, se ha decidido definir una serie de objetivos parciales:

- **Estudio detallado de las herramientas a emplear en el desarrollo de la aplicación:** Dado que ya se poseen conocimientos sobre el lenguaje de desarrollo de Android, tan solo se deberá hacer hincapié en la investigación y estudio de las herramientas que permitan la captura y manejo de imágenes en Android así como la creación de documentos donde renderizar e introducir dichas imágenes.
- **Estudio de las tecnologías a emplear para el desarrollo de los servicios Web y configuración del servidor:** Los documentos e información de la aplicación serán almacenadas en un servidor, se deberán investigar las posibles configuraciones del mismo, así como las tecnologías o métodos que nos permitan implementar los servicios Web.

- **Estudio de técnicas innovadoras de interacción con las aplicaciones:** Para ofrecer la mejor experiencia al consumidor, se estudiarán formas de interacción (logueo, creación de usuarios, navegación por el sistema...) que en conjunto transmitan sencillez, innovación, eficacia y confort al usuario.
- **Valoración de alternativas para alojar la información y los documentos:** Con el fin de proporcionar una herramienta única que combine el escaneado con el almacenamiento de información, se deberán estudiar las alternativas disponibles en el mercado actual que satisfagan los requisitos especificados y resulten opciones viables económicamente.
- **Diseño de la aplicación:** Una vez adquirido el conocimiento necesario, se podrá realizar el diseño detallado del sistema a desarrollar.
- **Implementación de los servicios Web**
- **Implementación de la aplicación**
- **Evaluación del sistema desarrollado:** Mediante el uso de pruebas de caja blanca, caja negra y pruebas con usuarios reales.

1.3 Planificación del proyecto

Para la implementación del proyecto, se ha seguido un **marco de desarrollo ágil** de software denominado **Scrum** [3], cuya estrategia de desarrollo es incremental. Siguiendo este modelo de desarrollo, se han delimitado periodos de dos semanas denominados **Sprints** al final de los cuales queda completada la creación de nuevos incrementos de software utilizable y que cumplen con requisitos previamente determinados para el Sprint correspondiente (Figura 3).

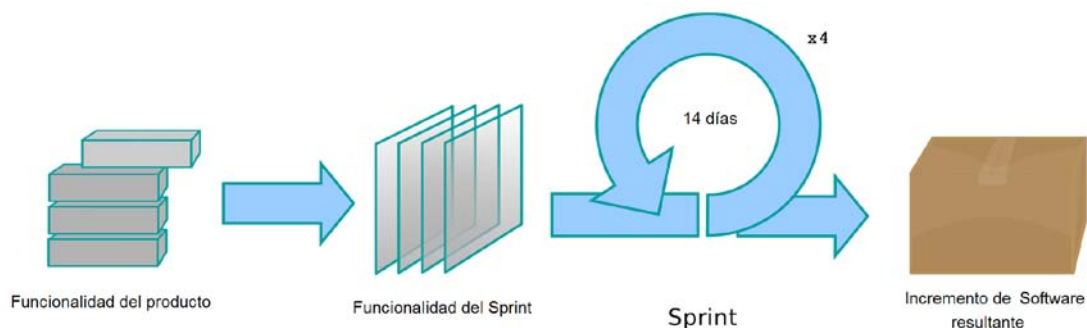


Figura 3. Marco de desarrollo ágil empleado (4 sprints). Fuente: Propia.

En total se ha dividido el desarrollo de la aplicación en cuatro **Sprints**, con un tiempo total de implementación de 8 semanas. La planificación en detalle del proyecto se puede encontrar en el **Diagrama de Gantt** adjunto al documento (**Anexo A**).

1.4 Estructura de la memoria

La memoria del proyecto se divide a lo largo del documento en los siguientes capítulos:

- **Capítulo 1.** Introducción.
- **Capítulo 2.** Estado del arte.
- **Capítulo 3.** Desarrollo del sistema.
- **Capítulo 4.** Evaluación del sistema.
- **Capítulo 5.** Conclusiones.

2. Estado del arte

En este capítulo, se ha llevado a cabo un análisis e investigación previa de los elementos que juegan un papel importante en el sistema que se va a desarrollar. A lo largo del apartado se exponen motivos por los que se ha seleccionado el sistema operativo **Android** y se analizan los resultados de la investigación de las herramientas y **métodos de escaneado, almacenamiento en servidores y compartición de documentos** existentes en la actualidad. Se muestra la información recogida en torno a diferentes **estándares de codificación** empleados hoy en día, los cuales nos permitirán, tras ser aprobada previamente esta funcionalidad en el siguiente capítulo, introducir un sistema innovador de autenticación de usuarios. Por último, se muestra un **análisis del mercado de aplicaciones** actual, en relación al proyecto en el que se va a trabajar.

2.1 Android

El objetivo del proyecto es alcanzar un medio de digitalización de documentos y de acceso a un **sistema de almacenamiento lo más simple y barato posible**. Tal y como se reseñó antes, los móviles son una buena alternativa. Dentro de los móviles inteligentes, los más baratos son los de tipo **Android**.

Poco tiempo después de su lanzamiento en 2008 [5], el sistema operativo multiplataforma, libre y gratuito, basado en el Kernel de Linux llamado Android, **conquistó el mercado de *smartphones*** y tablets debido a su atractivo aspecto y su eficiente trabajo. En tan solo tres años, se posicionó líder de ventas de teléfonos inteligentes en el mercado, consolidando esa posición en los años posteriores y manteniéndola hoy en día [4] (Figura 4).

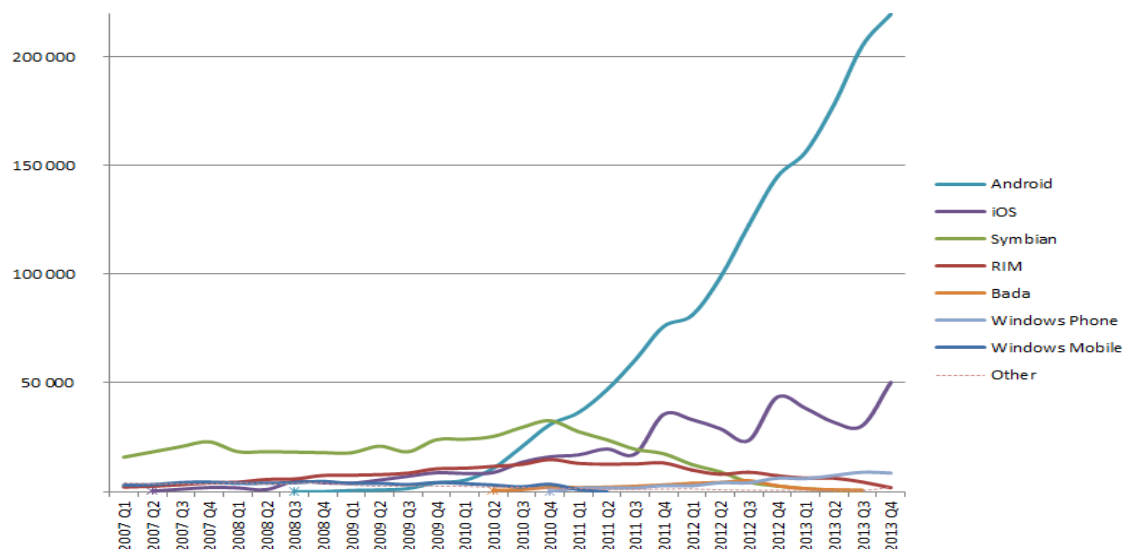


Figura 4. Ventas mundiales de Smartphones. Fuente:Wikipedia.

Tras considerar el desarrollo de la aplicación en otros sistemas operativos como iOS o Windows phone, y pese a que las cifras de ventas varíen entre los diferentes trimestres [tabla 1], el **éxito de Android** nos proporciona la seguridad de haber elegido un sistema operativo robusto, en continua evolución y que día a día seduce más usuarios.

Trimestre	Android	iOS	Windows Phone
2014 Q4	76.60%	19.70%	2.80%
2014 Q3	84.37%	11.70%	2.90%

Tabla 1. Variación de ventas trimestrales de dispositivos móviles *smartphone*

Una vez seleccionado el sistema operativo objetivo, es necesario estudiar las versiones existentes de éste y evaluar para cuales sería conveniente desarrollar una aplicación compatible.

Considerando la figura expuesta a continuación (Figura 5), se ha decidido implementar el sistema para **versiones 4.0** en adelante, dando de esta forma soporte al **90.4% de los usuarios**. En efecto, esta decisión implica que no se daría soporte a un 10% del mercado total de Android. Esto constituye una **opción razonable**, sobre todo si se tiene en cuenta que las primeras versiones de la API Android presentan ciertos problemas a la hora del desarrollo de determinadas funcionalidades.

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
2.2 Froyo	8	99,5%
2.3 Gingerbread	10	90,4%
4.0 Ice Cream Sandwich	15	82,6%
4.1 Jelly Bean	16	61,3%
4.2 Jelly Bean	17	40,9%
4.3 Jelly Bean	18	33,9%
4.4 KitKat	19	< 0.1%
5.0 Lollipop	21	

Figura 5. Versiones de Android y su porcentaje de distribución. Fuente: AndroidStudio.

2.2 Digitalización de documentos

La digitalización de documentos o *document imaging* es una tecnología de la información para sistemas, capaz de replicar documentos y convertirlos a imágenes digitales.

El **pantelégrafo** (Figura 6) es el predecesor de los escáneres existentes hoy en día. Se servía de electroimanes para sincronizar el movimiento de un péndulo en el origen y en el destino, siendo capaz de replicar escritura a mano, firmas e imágenes [6].

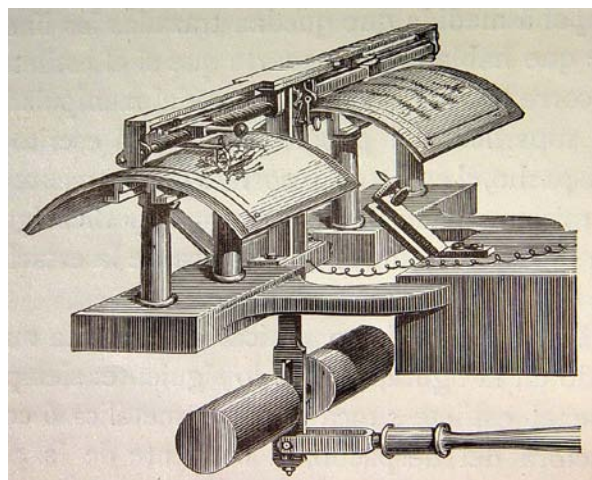


Figura 6. Pantelégrafo Fuente: Flickr.

Existen una gran variedad de escáneres en la actualidad que pueden cubrir todo tipo de necesidades de digitalización, no obstante presentan inconvenientes a la hora de transportarlos y hacer uso de ellos en cualquier contexto. La fabricación de **escáneres portátiles** hizo frente a este problema, pero la alta precisión necesaria en su manejo para obtener resultados óptimos convierte esta herramienta en una opción poco atractiva. Posteriormente se fabricaron **escáneres 3D** que solventan los problemas de transporte y de precisión requerida, pero son una opción inviable para el uso del usuario corriente, bien por su elevado precio o por su funcionalidad orientada al usuario profesional. En cualquier caso, **no es habitual** que las personas **se desplacen transportando un escáner** y, por tanto, es razonable plantear alguna alternativa al escáner mediante un medio tecnológico con el que los usuarios cuentan habitualmente.

La solución al problema de encontrar una herramienta fácilmente portable y que permita el escaneado de documentos en cualquier situación viene dada por el **uso de los smartphones** como instrumento de escaneado. De hecho no es difícil encontrar soluciones software para utilizar este potencial de los *smartphones*. Algunos ejemplos de software que aborda este problema y proponen un resultado eficiente son [7]:

- **Google Drive** (Android/iOS). Permite la **digitalización mediante cámara**, pero la **información se almacena inevitablemente en Google Drive**.
- **CamScanner** (Android/iOS/Windows Phone). No tiene integrado la funcionalidad de subida de los datos a servidor de almacenamiento.
- **Scannable by Evernote** (iOS). No tiene integrado la funcionalidad de subida de los datos a servidor de almacenamiento.
- **FineScanner** (iOS). No tiene integrado la funcionalidad de subida de los datos a servidor de almacenamiento.

El software que se pretende diseñar, debe **separar el cliente de digitalización del servidor de almacenamiento**. Además, se quiere evitar la dependencia respecto a un único medio de almacenamiento, por lo que la base de datos del *backend* a diseñar sólo incluirá rutas de acceso a los archivos digitales, permitiendo que los mismos puedan alojarse en diversos emplazamientos.

2.3 Almacenamiento de datos en servidor

Cuando se habla de almacenar datos en un servidor, se está haciendo referencia a un modelo de almacenamiento de datos basado en redes denominado **almacenamiento en nube** o *cloud storage* [9].

El **almacenamiento en nube** consiste en la custodia de datos en un sistema de almacenamiento externo mantenido por un tercero. En lugar de almacenar la información en el disco duro del ordenador del usuario o en otro dispositivo de almacenamiento local, lo guarda en una **base de datos remota** que establece una conexión con el ordenador haciendo uso de Internet [8].

El empleo del *cloud storage* nos permitirá aprovechar las **ventajas** intrínsecas a este tipo de modelo de almacenamiento y adaptarlas a nuestro proyecto. Usando un modelo de almacenamiento local el usuario vería limitado el acceso a su información en caso de no poseer el dispositivo donde se ha producido el escaneado, por el contrario, utilizando *cloud storage* el usuario podrá **acceder a su cuenta y documentos desde cualquier dispositivo** que cuente con la aplicación. Además la elección de un sistema de almacenamiento adecuado, nos permitirá **compartir los datos con otros usuarios**, siendo éste un objetivo del sistema a desarrollar.

A día de hoy, existen numerosas empresas especializadas que ofrecen interesantes planes de almacenamiento de datos en la nube. Algunos servicios de *cloud storage* son:

- **Dropbox:** Da soporte a Linux y Blackberry además de los normales estándares Windows, Mac OS X, Android y iOS. Dropbox es una solución multiplataforma excelente, que sigue siendo un punto de referencia a la hora de escoger un servicio de almacenamiento en nube.
- **Google Drive:** Dispone al usuario de 15Gb de almacenamiento tan solo por configurar su cuenta. La interfaz de uso de la app es inteligente y sencilla de emplear.
- **Mega:** Garantiza a los usuarios una total privacidad y seguridad al proporcionar una encriptación en cada parte del proceso de almacenamiento (local, en ruta y en el servidor). Con su generosa cuenta gratuita, servicio rápido y su naturaleza de alta seguridad, Mega es una muy buena opción para la mayoría de las personas que buscan una solución de almacenamiento en línea.
- **Copy:** Una alternativa decente a Dropbox y OneDrive, pues ofrece una generosa cantidad de espacio gratuito, pero en realidad son los servicios dedicados a las empresas lo que hacen a este servicio interesante.
- **One Drive:** Sus recientes actualizaciones han convertido este servicio en una opción competitiva, sobre todo ahora que ofrece 15 GB de espacio gratuito.
- **Hosting:** El *hosting* en servidor es otra forma de sistema de almacenamiento. La ventaja de emplear un servidor propio es la posibilidad de emplear otros sistemas de almacenamiento dentro de éste.

Finalmente, tras haber analizado el mercado actual de servicios de *cloud storage*, se ha decidido emplear el *hosting* en un servidor bajo nuestro control. La finalidad de esta elección es la de poder proporcionar **un servicio adaptado a las necesidades de la aplicación** y la de aprovechar la ocasión para adquirir conocimiento extra en un tema tan importante e interesante que es desconocido para mí.

2.4 Compartición de documentos

Se denomina compartición de documentos o *file sharing*, a la práctica de proporcionar acceso o distribuir medios digitales a otros usuarios.

Existen dos métodos de compartición de documentos en la actualidad [10]:

- **Compartición de archivos peer-to-peer:** Consiste en el empleo de software especializado que se conecta a una red peer-to-peer (P2P) para buscar archivos compartidos en los equipos de otros usuarios conectados a la red y descargar los archivos directamente de estos usuarios. Normalmente, los archivos grandes son divididos en **partes más pequeñas** las cuales, una vez finalizada la compartición, volverán a ser unidas por el software que ha llevado a cabo la descarga. El programa de descargas P2P más conocido es BitTorrent.
- **Sincronización de archivos y servicios de compartición:** Consiste en la creación de directorios en los ordenadores o dispositivos móviles de los usuarios, que serán sincronizados para formar un solo directorio al que estos podrán acceder independientemente del dispositivo empleado para ello. De esta forma se puede tener acceso fácilmente (desde páginas webs, aplicaciones móviles...) a los ficheros alojados en estos directorios, permitiendo así su compartición.

Para la herramienta a desarrollar, se ha decidido emplear el segundo método explicado. Se implementará un **servicio de sincronización tanto en el servidor como en la aplicación** con el fin de cumplir el objetivo de compartir documentos entre usuarios (Figura 7).

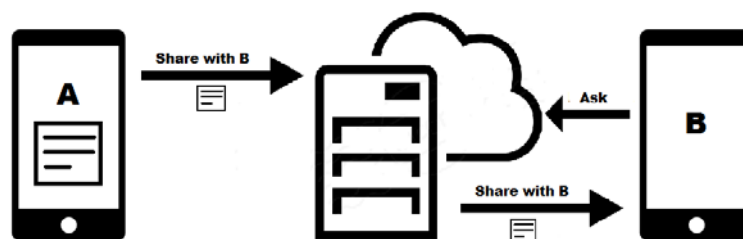


Figura 7. File sharing empleado. Fuente: Propia.

2.5 Códigos digitales para la identificación de recursos

Persiguiendo el objetivo de mejorar la experiencia de uso de la aplicación mediante **técnicas innovadoras**, se ha decidido incorporar una funcionalidad de **ingreso (login)** a la plataforma por medio de **códigos personalizados** para cada usuario. Posteriormente, esos códigos deberán ser empleados para la **identificación del autor** frente a un administrador a la hora de **registrar sus documentos**. De este modo, en esta propuesta se presentará un **sistema alternativo** al sistema convencional de autenticación basado en el par nombre de usuario/contraseña.

Entre los códigos más empleados en la actualidad figuran los **códigos de barras** y los **bokodes**. Los bokodes son pequeños diodos LED recubiertos de un material con condiciones ópticas especiales y que presentan diversas ventajas, como su capacidad de almacenamiento de cientos de bits y su menor tamaño [11], pero el hecho de necesitar implementación física lo hace inviable para ser generado por nuestra aplicación. Por ese motivo se emplearán **códigos de barras**.

Existen tres grandes tipos de códigos de barras destinados a usos específicos [12], estos son **códigos de barras numéricos**, **códigos de barras alfa-numéricos** y **códigos bidimensionales**. Cada uno de estos tipos está formado por múltiples estándares que se diferencian entre sí por la cantidad de datos que pueden almacenar, el método de codificación, el método para calcular la suma de verificación (checksum) del código, etc.

Tras realizar una evaluación de los datos que necesita almacenar nuestra aplicación e investigando cual es el código que presenta mayor facilidad para ser escaneado por un *smartphone*, se ha seleccionado la **codificación bidimensional QR** como estándar de codificación de usuario en nuestro sistema (Figura 8).



Figura 8. Código QR.
Fuente: Propia.

2.6 Mercado actual

Tras las investigaciones realizadas a lo largo del **capítulo 2**, se ha podido observar la existencia de aplicaciones en el extenso mercado de Android que cumplen funcionalidades que nuestro proyecto persigue. Un claro ejemplo son **DropBox** y **Google Drive**.

El hecho de introducir en nuestro proyecto una herramienta innovadora de autenticación mediante códigos de barras, nos va a permitir alejarnos de la propuesta deparada por las dos aplicaciones mencionadas. Actualmente **no existe en el mercado**

ninguna aplicación que combine el escaneado, almacenamiento y compartición de documentos, con un sistema de *login* e **identificación de autoría por medio de códigos personalizados de usuario**. Además, como se ha mencionado previamente, nos interesa una herramienta que **permita escalabilidad en cuanto a medios de almacenamiento se refiere**, tomando como ejemplo los modelos de nube híbrida [18].

3. Desarrollo del sistema

3.1 Análisis

Con el fin de asegurar el cumplimiento de todos los objetivos del sistema planteado, en primer lugar se realizará un **análisis** de los aspectos importantes a tener en cuenta para su posterior diseño e implementación. Este análisis se dividirá en un estudio de los **usuarios finales** de la aplicación y lo que esperan de ésta, un análisis de **requisitos del sistema** y finalmente un análisis de las **herramientas y tecnologías** que se emplearán para cumplir los requisitos especificados.

3.1.1 Análisis de usuarios

Se definen como usuarios del sistema al conjunto de individuos que van a utilizar dicho sistema y que van a hacer uso de su funcionalidad, de forma regular o esporádica [13]. La evaluación de los usuarios de la herramienta nos permitirá posteriormente realizar un mejor análisis y, consecuentemente, definir los requisitos de nuestra aplicación de acuerdo con opiniones y expectativas de estos individuos.

El objetivo del sistema es que sea empleado por **usuarios** que precisen de éste independientemente de la edad, existiendo un amplio rango de edades a tener en cuenta. Este rango de cifras aumenta al ser conscientes de que, actualmente de media, los niños reciben un *smartphone* a la edad de 11 años [19].

Finalmente se ha decidido estimar que el rango de edades de los usuarios puede ser desde los **14 años hasta los 80 años** aproximadamente. Se ha realizado una encuesta a representantes de diferentes edades para conocer lo que les **atrae de una aplicación móvil**, para que nos ayuden con aspectos del **diseño** y de paso para conocer cómo **reaccionarían ante una funcionalidad** que se ha propuesto como forma de innovar el proceso de *login*.

Nombre	Edad	¿Qué debe tener una buena app?	¿Qué color o combinación de colores recomendarías?	¿Qué te parecería poder realizar el logueo mediante la cámara en lugar de usar el <i>login</i> password?
Iñaki	15	Fácil de usar Gratuita	Blanca y negra	Ahorraría tiempo
Myriam	27	De confianza Que respete la privacidad Que no sea necesario registrarse Gratuita Rápida Que no consuma muchos recursos	Una combinación que contraste, negro o blanco con algún color vivo	Me parece una idea innovadora y que evita tener que usar el teclado lo que la hace más rápida
Maite	54	Fácil de usar Segura Gratuita	Un color cálido	- - -
D. Plaza	62	Que no pida mucha información Fácil de usar	- - -	- - -

Tabla 2. Encuesta a los usuarios regulares

Aunque las respuestas obtenidas fueron variadas, una de ellas fue casi unánime entre todos los usuarios encuestados. En efecto, una de las cosas que más valoran es el precio de la aplicación. Ante este resultado se les cuestionó si estaban dispuestos a **perder** cierta **seguridad en el sistema a cambio de abaratar la aplicación**, a lo que contestaron afirmativamente.

Por otra parte la encuesta demostró que el **método innovador de login** tiene **aceptación entre el usuario regular** y en caso de no preferir esta técnica siempre se podrá recurrir al par nombre de usuario/contraseña habituales.

3.1.2 Análisis de requisitos

En esta sección se recogen los requisitos que presenta el sistema. Se consideran requisitos de la aplicación a las necesidades documentadas del comportamiento o funcionalidad esperada del producto. Los requisitos se dividirán en **funcionales** (funcionalidades concretas a implementar por la aplicación) y **no funcionales** (características de la aplicación que señalan una restricción de la misma).

3.1.2.1 Requisitos funcionales

En esta sección se detallarán los Requisitos Funcionales (RF) y se dividirán en subsistemas atendiendo a su tipo de funcionalidad que desempeñará en la aplicación.

Gestión de usuarios

RF (01) Registrar usuarios: Se podrá dar de alta a un nuevo usuario introduciendo el nombre de usuario, *password*, nombre propio, apellido y la dirección de correo electrónico. Tras enviar los datos al servidor y realizar las comprobaciones pertinentes, el usuario quedará registrado en la base de datos con su nombre de usuario como clave primaria.

RF (02) Autogenerar código personal: Como parte del proceso de creación de cuenta, se autogenerará un código personalizado para el usuario atendiendo a la información introducida en el formulario de registro.

RF (03) Registrar código personal: Durante el proceso de creación de cuenta de usuario, éste tendrá la opción de escanear un código (QR) de su elección. Tras ser escaneado, se comprobará su disponibilidad con el servidor y en caso de estar disponible le será asignado como código personal.

RF (04) Realizar el *login* en la aplicación mediante nombre de usuario y contraseña: El usuario podrá identificarse para acceder al contenido de la aplicación empleando para ello su nombre de usuario y la contraseña introducida en el formulario de registro. Tras realizar la comprobación de los datos con el servidor (y en caso de que el *login* sea correcto), el usuario será redirigido a su pantalla de inicio en la aplicación.

RF (05) Realizar el *login* en la aplicación mediante el código personalizado: El usuario podrá identificarse para acceder al contenido de la aplicación empleando para ello su código personal. Tras escanear el código, la aplicación lo decodificará y recuperará la información del usuario del servidor, redirigiendo a la pantalla de inicio del usuario en la aplicación.

RF (06) Acceder al código personalizado del usuario: Una vez identificado en la plataforma, el usuario podrá acceder en todo momento a su código personalizado, el cual será impreso en la pantalla del dispositivo móvil que esté corriendo la aplicación.

RF (07) Realizar *logout* de la aplicación: Una vez identificado en la plataforma, el usuario podrá salir de su sesión conservando en todo momento sus documentos almacenados en el servidor.

Gestión de contraseñas

RF (08) Recuperar la contraseña introduciendo el mail: El usuario tendrá la posibilidad de recuperar su contraseña en caso de pérdida, introduciendo el mail asociado a su cuenta en la aplicación. La aplicación contactará con el servidor para recuperar la password y enviarla al correo del usuario que lo ha solicitado.

RF (09) Recuperar la contraseña escaneando el código personal: El usuario tendrá la posibilidad de recuperar su contraseña en caso de pérdida, escaneando su código personalizado. La aplicación rescatará los datos asociados a ese código y mostrará por pantalla el nombre de usuario y la contraseña.

Gestión de documentos

RF (10) Realizar el escaneado de documentos: El usuario que ha sido autenticado será capaz de emplear la herramienta para crear un documento a partir de fotografías realizadas con el dispositivo. Para generar el documento será necesario introducir el nombre del mismo y el número de páginas a escanear. Una vez realizadas las fotografías del documento físico, la aplicación generará el fichero pdf y lo almacenará en el servidor.

RF (11) Clasificar documentos mediante etiquetas: El usuario autenticado podrá introducir etiquetas a la hora de crear los archivos con el fin de clasificarlos posteriormente.

RF (12) Acceder a la librería de documentos: El usuario autenticado podrá acceder a su repositorio de archivos donde se listarán los documentos de su autoría. Desde esta librería podrá seleccionar el documento que le interese y le será mostrada su información (etiquetas, autor y usuarios con los que está compartido el documento).

RF (13) Descargar documentos de la librería: El usuario autenticado podrá seleccionar un documento de su librería y descargárselo a su *smartphone*. La aplicación obtendrá del servidor la URL donde se aloja dicho documento.

RF (14) Eliminar documentos de la librería: El usuario autenticado podrá seleccionar un documento de su librería y eliminarlo de ésta. La aplicación realizará los trámites necesarios de actualización de base de datos y de liberación del espacio que ocupaba el documento.

RF (15) Compartición de documentos de forma pública: El usuario autenticado podrá compartir documentos con todos los miembros de su grupo sin excepciones. Dicho documento pasará a ser público en el grupo y podrá ser

descargado por todos los usuarios con los que se haya compartido, no obstante solo podrá ser eliminado por el usuario que haya realizado la compartición.

RF (16) Compartición de documentos de forma privada: El usuario autenticado podrá compartir documentos con un miembro en particular del grupo, para ello deberá escanear el código personalizado de dicho usuario. El documento compartido podrá ser descargado por el usuario al que se le ha dado acceso, pero éste no tendrá permiso para eliminarlo.

RF (17) Acceder a lista de ficheros del grupo: El usuario autenticado podrá acceder a la lista e información de los ficheros de los grupos de los que forma parte y que estén compartidos con él. En caso de ser administrador del grupo tendrá permisos de lectura y de escritura sobre estos documentos.

RF (18) Descargar documentos compartidos en el grupo: El usuario autenticado podrá descargar a su dispositivo *Smartphone* los documentos de los grupos de los que forma parte y que cumplan la condición de estar compartidos con él.

RF (19) Eliminar documentos compartidos en el grupo: El usuario autenticado podrá eliminar aquellos documentos que estén compartidos en un grupo del cual sea administrador. La aplicación actualizará las bases de datos y liberará el espacio ocupado por dicho documento en el servidor.

Gestión de grupos

RF (20) Crear grupo: El usuario autenticado como administrador podrá crear un grupo. Se deberá introducir el nombre del grupo y unas etiquetas para su clasificación.

RF (21) Añadir miembros al grupo: El usuario autenticado como administrador podrá añadir usuarios al grupo que acaba de crear, para ello deberá escanear los códigos personalizados de estos usuarios.

RF (22) Salir del grupo: El usuario autenticado podrá salir de cualquier grupo del que sea miembro. La aplicación eliminará dicho grupo de su lista de grupos y le retirará el acceso al mismo.

RF (23) Eliminar grupo creado: El usuario autenticado como administrador podrá eliminar cualquier grupo que haya creado. La aplicación actualizará la base de datos y eliminará al grupo del sistema.

RF (24) Acceder a la información del grupo: Cualquier usuario que forme parte de un grupo, ya sea miembro normal o administrador, podrá acceder a la información de dicho grupo.

RF (25) Acceder a lista de miembros del grupo: El usuario autenticado y administrador de un grupo, podrá acceder a la lista de usuarios que forman parte de dicho grupo.

RF (26) Excluir un usuario de un grupo: El usuario autenticado y con rol de administrador de un grupo, podrá bloquear a cualquier usuario del grupo, impidiendo así su acceso tanto a la información como a los documentos del mismo. La aplicación actualizará la base de datos del servidor.

Gestión de búsqueda

RF (27) Buscar documentos: Cualquier usuario autenticado en la aplicación podrá realizar búsquedas de sus documentos, en las pantallas del sistema que así lo permitan (aquellas que listen los documentos), aplicando filtros sobre el nombre o las etiquetas.

RF (28) Buscar usuarios: Los usuarios administradores de grupos podrán realizar búsquedas de los miembros de dichos grupos, en las pantallas del sistema que así lo permitan (aquellas que listen los miembros de un grupo), aplicando filtros sobre el nombre de estos usuarios.

RF (29) Buscar grupos: Cualquier usuario autenticado en la aplicación podrá realizar búsquedas de los grupos a los que pertenece, tanto como miembro o como administrador, aplicando filtros sobre el nombre o las etiquetas de dichos grupos.

RF (30) Actualizar listas en tiempo real: Todas las pantallas que listen tanto documentos, como usuarios o grupos, serán actualizadas con los datos del servidor en tiempo real.

3.1.2.2 Requisitos no funcionales

Tras definir las funcionalidades que debe cumplir el sistema, en este apartado se especificarán los requisitos que debe cumplir la aplicación para asegurar una calidad máxima de la aplicación y unas prestaciones que satisfagan al usuario.

RNF (01) Mantenibilidad: Con la mirada puesta en futuras modificaciones sobre el proyecto, éste debe ser fácil de mantener. El código estará debidamente

comentado y se perseguirá minimizar el acoplamiento entre componentes y maximizar la cohesión.

RNF (02) Usabilidad: La aplicación debe ser utilizada sin problemas por todos los usuarios poseedores de un *smartphone*. Las instrucciones deben ser concisas y la interacción con las diferentes pantallas de la herramienta debe ser intuitiva.

RNF (03) Portabilidad: No se puede esperar que todos los usuarios operen sobre el mismo *smartphone*. La aplicación debe ejecutarse sin problemas en todo tipo de dispositivos que cuenten con un sistema operativo Android 4.0 o superior.

RNF (04) Robustez: La aplicación debe presentar una sólida gestión de excepciones y situaciones imprevistas que puedan ocurrir. La aplicación no debe intentar operar con el servidor si no posee una conexión a la red, y debe salvaguardar los documentos en caso de haber un fallo.

RNF (05) Rendimiento: Para comodidad del usuario, la aplicación debe presentar una interacción rápida con el servidor, las peticiones no se resolverán en menos de 2 segundos. No se asegura una velocidad de descarga de documentos elevada pues depende de la red a la que esté conectado el usuario. Además la aplicación no llenará de archivos extra (como imágenes o documentos) la memoria del dispositivo desde donde opere.

RNF (06) Coste: El coste de desarrollo de la aplicación y de las tecnologías de las que se hace uso no deben sobrepasar un presupuesto excesivamente limitado.

3.1.3 Análisis de herramientas y tecnologías

Para el desarrollo de un sistema que cumpla los requisitos especificados en los apartados anteriores, se va a precisar del uso de diferentes herramientas y tecnologías. En este apartado se listan y describen cada una de ellas.

- **Android Studio:** Para el desarrollo de la aplicación, se empleará el entorno de desarrollo integrado oficial (IDE) de Android. Esta herramienta destaca por características como su **renderización en tiempo real**, **refactorización** específica de Android o su opción de **pre visualizado** en múltiples configuraciones de pantalla.
- **Android SDK:** Se emplearán herramientas del kit de desarrollo de software de Android entre las que destaca el **debugger**, mediante el cual se depurará el código de la aplicación en las situaciones que así lo precisen (encontrar fallos, testear requisitos...).
- **Java:** Para la programación de la aplicación de Android se ha elegido este lenguaje orientado a objetos y de propósito general.
- **PHP:** Para el desarrollo de la API se ha seleccionado este lenguaje de programación de uso general, de código del lado del servidor y diseñado para el desarrollo web de contenido dinámico
- **BitBucket:** Se ha elegido realizar el proyecto en un marco de desarrollo ágil, lo que implica cambios constantes en el código que implementan nuevas funcionalidades para el Sprint. Este acelerado ritmo de desarrollo exige tener una herramienta donde mantener un repositorio del proyecto para poder acceder a los cambios realizados y a las múltiples versiones del mismo. Se ha seleccionado este servicio de **hosting basado en web**, que cubre las necesidades descritas.
- **Hosting:** Para poder cumplir los requisitos de almacenamiento y compartición de documentos, y para hacer de la aplicación una plataforma accesible en cualquier emplazamiento, se precisa de un servidor donde alojar la API y la base de datos (i.e., el *backend* de nuestra aplicación) contra la que trabajará el cliente Android (nuestro *frontend*). Se ha especificado que el presupuesto a emplear en el desarrollo de la aplicación y sus herramientas es limitado, por lo que se descarta el alquiler de un servidor dedicado. Para adaptarse al **RNF06** se opta por un servicio de *hosting* gratuito que suple sus carencias de seguridad (no da soporte a protocolo SFTP o conexiones SSL) con su bajo coste. El *hosting* seleccionado es **000webhost.com**.

- **MySQL:** Para la implementación de la **base de datos de la aplicación**, se empleará este sistema de administración de bases de datos relacional, multihilo y multiusuario. Este sistema se sirve de múltiples tablas para almacenar y organizar la información y se adapta a diferentes entornos de desarrollo, permitiendo la interacción con el lenguaje seleccionado para la implementación de la API (PHP).
- **PhpMyAdmin:** Se ha seleccionado esta herramienta con el fin de mejorar la administración de la herramienta anterior (MySQL) a través de la web. PhpMyAdmin proporciona una interfaz de usuario con la que introducir las operaciones necesarias para administrar la base de datos del sistema.
- **ZXing:** Con el fin de cumplir los requisitos de escaneo y generación de códigos personalizados (RF03, RF05, RF06, RF09, RF16 y RF21), se ha hecho uso de esta librería Android que, una vez integrada en la aplicación, nos evita tener que recurrir a aplicaciones externas que realicen el escaneo.
- **Itextpdf:** Con el fin de cumplir el requisito de generación de pdf (RF10), se ha hecho uso de esta librería Java que permite la creación programática de documentos.
- **Ftp4j:** Se ha empleado esta librería java que proporciona un cliente FTP con el que se transferirán los documentos generados por los usuarios, al servidor donde serán almacenados.
- **JavaMail:** Con el fin de cumplir el requisito de recuperación de la contraseña por email (RF08), se ha hecho uso de esta librería Java que permite la configuración y uso de un cliente de correo electrónico de forma programática.

3.2 Diseño

Una vez finalizado el análisis del proyecto a desarrollar, el siguiente paso es definir la **arquitectura, lógica, interfaces y estructuras de datos** que nos permitirán satisfacer los requisitos especificados. Al finalizar esta sección, se contará con un diseño que permitirá comenzar con el proceso de implementación del sistema.

3.2.1 Diseño de la arquitectura del sistema

Para la correcta implementación del sistema se debe elegir una arquitectura cuyas ideas se basen en la reutilización del código y en la separación de conceptos, que se traducen en una **máxima cohesión** y un **mínimo acoplamiento**. Estas características permitirán facilitar la tarea de desarrollo de la aplicación y su futuro mantenimiento.

El patrón de arquitectura software que persigue estos ideales es el **Modelo Vista Controlador** (MVC). Este patrón se caracteriza por la división del software de la aplicación en tres partes interconectadas. Eliminando las dependencias innecesarias entre los componentes al implementar el software, da lugar a un código con menos errores y más fácil de mantener pues es reutilizable sin modificaciones [14]. Los tres pilares fundamentales del patrón elegido son:

- **Modelo:** gestiona directamente los datos, la lógica y las reglas de la aplicación.
- **Vista:** representa la salida de información de la aplicación, pudiendo tomar forma de interfaz de usuario, gráficos, tablas...
- **Controlador:** acepta la entrada de comandos permitiendo la interacción entre el modelo y la vista.

La plausibilidad del patrón **MVC** en Android no es del todo aceptada, estando este reparo motivado por la imposibilidad de instanciar las **actividades** (Controlador) de la aplicación sin tener en cuenta el **layout** de las mismas (Vistas). En efecto, de acuerdo con esta consideración, quedaríamos abocados a un MVC **altamente acoplado**. Ahora bien, se puede optar por implementar de modo independiente los **controladores y las clases que cargarán los layout**. Así, se intentan subsanar los acoplamientos entre la Vista y el Controlador del diseño, lográndose una mayor independencia respecto a los controladores [15]. De esta forma, el MVC personalizado (Figura 9) para este proyecto contará con los siguientes componentes:

- **Modelo:** Lo formarán las **clases java que implementan los objetos** de la aplicación (datos y lógica), así como toda aquella información que se almacenará en **la base de datos del servidor** destinado para este fin.
- **Vista:** Lo formarán los **ficheros XML** que contendrán tanto el **layout** de la aplicación, como las **clases java donde irán incrustados dichos ficheros**. Estas clases permitirán a los usuarios interactuar con la aplicación y a través de las mismas se realizarán todas las llamadas al controlador permaneciendo desacopladas de éste.
- **Controlador:** Lo formarán las **clases java** donde se implementarán, las **interacciones de la vista con el modelo** y la posterior **actualización de información** tanto en la vista pertinente como en dicho modelo en caso de ser necesario.

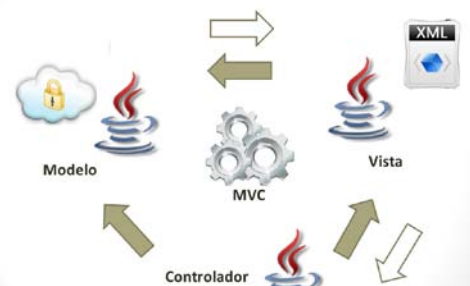


Figura 9. Patrón de arquitectura de software de la aplicación.

Fuente: Propia.

3.2.2 Diseño del modelo de datos del sistema

Resulta de vital importancia tener claro el **modelo** de datos que se empleará en el desarrollo de la aplicación.

En este apartado se definen y explican las **clases** que instanciarán a los elementos de la aplicación y las **relaciones de éstas**. También se profundiza en el **diseño de las estructuras de datos** destinadas a alojar la información de nuestra aplicación.

3.2.2.1 Elementos del sistema

Los elementos del sistema que forman el modelo de la aplicación son:

USER: Usuario base de la aplicación. La funcionalidad a la que accederá es: escaneado y almacenamiento de documentos, administración de su repositorio de archivos y acceso a los documentos compartidos de los grupos de los que es miembro. Los atributos de este elemento del sistema son su nombre de usuario, nombre de pila, apellido, contraseña, email y código personal.

ADMIN: Subclase que extiende de USER. Su funcionalidad amplía la del usuario base pudiendo crear grupos, administrarlos y compartir documentos con los miembros de dicho grupo.

FILE: Fichero escaneado y almacenado en el servidor por los usuarios del sistema. Podrán ser compartidos, eliminados y descargados desde la aplicación. Los atributos con los que cuentan los ficheros son su nombre, el autor, la url donde se aloja en el servidor y las etiquetas para clasificarlo.

GROUP: Grupos internos de la aplicación que serán creados y eliminados por los administradores correspondientes. Estos administradores podrán regular los miembros de los grupos y compartir documentos con los mismos. Los atributos con los que cuentan los grupos son el nombre y su administrador.

Una vez especificadas las clases con las que contará la aplicación, es necesario exponer las relaciones entre las mismas:

AUTHOR: Relación de composición entre la clase User y File, representa la posibilidad que tiene un usuario de ser el autor de 0 o N ficheros mientras que un documento solo puede poseer un autor.

SHARED (I): Relación de agregación entre la clase File y User, representa la posibilidad que tiene un fichero de ser compartido con 0 o N usuarios, y la posibilidad de un usuario de existir 0 o N ficheros compartidos con él.

SHARED (II): Relación de composición entre la clase Group y File, representa la posibilidad que tiene un grupo de tener compartidos de 0 a N ficheros, o la de un fichero de ser compartido o no con un grupo.

MEMBER: Relación de agregación entre la clase Group y User, representa la posibilidad que tiene un grupo de contar con 1 o N miembros, o la que tiene un usuario de pertenecer a 0 o N grupos.

OWNER: Relación de composición entre la clase Admin y Group, representa el número de 0 a N grupos de los que un administrador puede ser creador. También queda reflejada la obligación de un grupo de tener un administrador.

Todas las relaciones descritas quedan reflejadas en el diagrama de clases correspondiente:

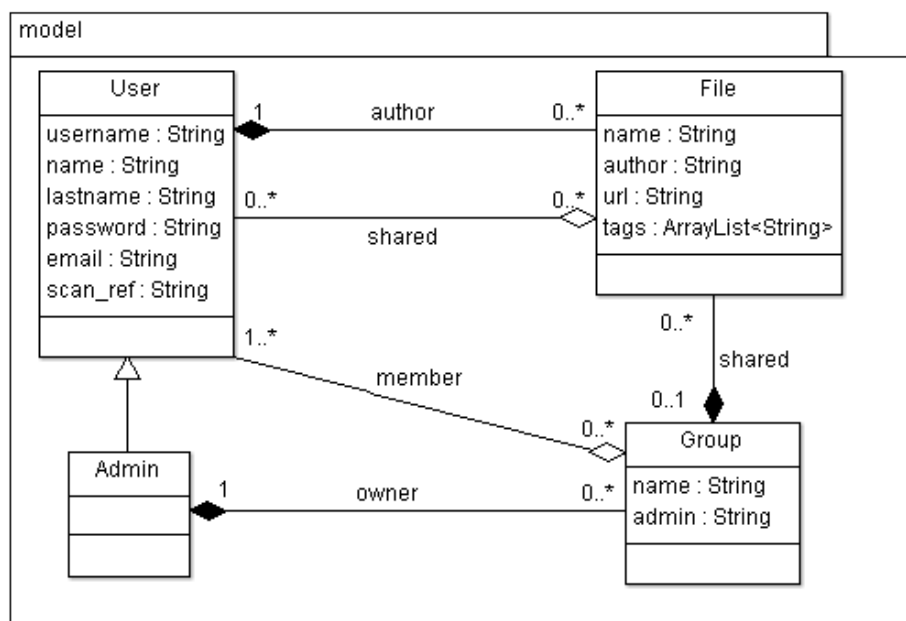


Figura 10. Diagrama de clases de la aplicación. Fuente: Propia.

3.2.2.2 Estructura de datos del sistema

A continuación se mostrará el diseño de las estructuras destinadas al almacenamiento de los datos dinámicos con los que trabajará la aplicación.

Tras analizar la funcionalidad del sistema se decidió estudiar los datos con los que trabaja la aplicación y se apreció una **ausencia** de requisitos que exijan operar con datos en local (en el móvil), eliminando la necesidad de alojar información en una **base de datos interna**. Además el carácter interactivo de la herramienta requiere

actualizaciones en tiempo real de los documentos y de la información de los grupos, **datos que son almacenados y recuperados del servidor**, por lo tanto todas las operaciones se realizarán sobre una **base de datos remota**.

La base de datos será relacional, se empleará el sistema MySQL para administrarla y las operaciones sobre ella serán gestionadas por funciones PHP. A continuación se muestra el diagrama Entidad-Relación del sistema a implementar (Figura 11).

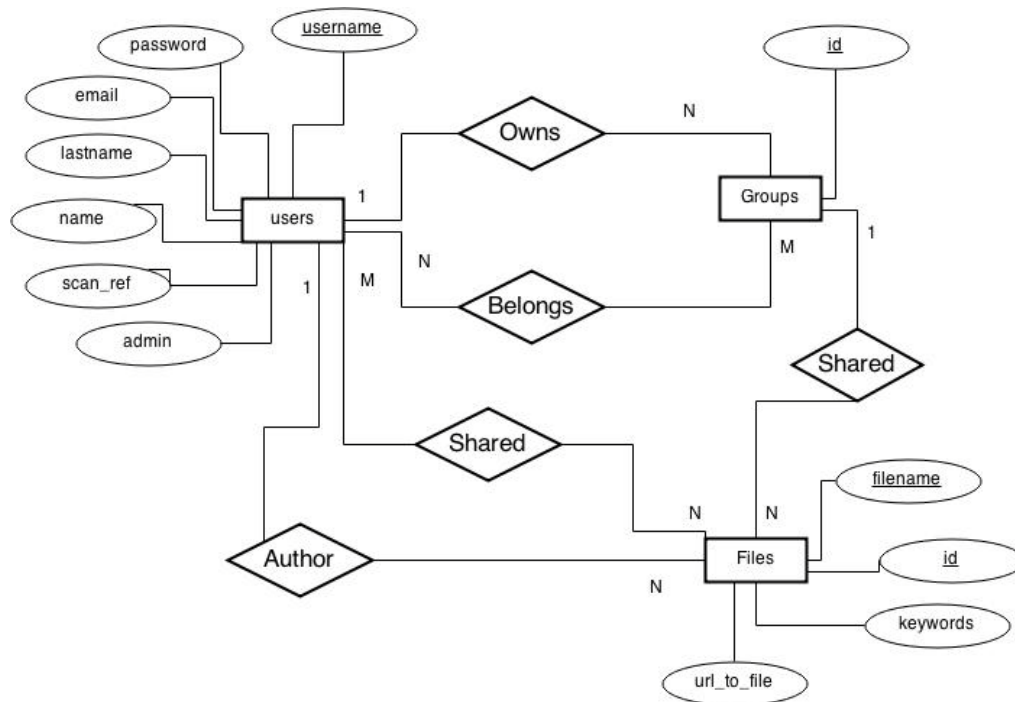


Figura 11. Diagrama ER. Fuente: Propia.

3.2.3 Diseño de la interfaz del sistema

Cuando hablamos de la **vista** de la aplicación, nos estamos refiriendo tanto a los ficheros XML que implementan las pantallas de interacción del sistema, como a las clases java que codifican las actividades en las que se cargan dichas pantallas.

Teniendo en cuenta la funcionalidad de la aplicación se ha diseñado un mapa de navegación que permite, de forma sencilla e intuitiva, interactuar con la herramienta y que cumple con el RNF02 de **usabilidad**.

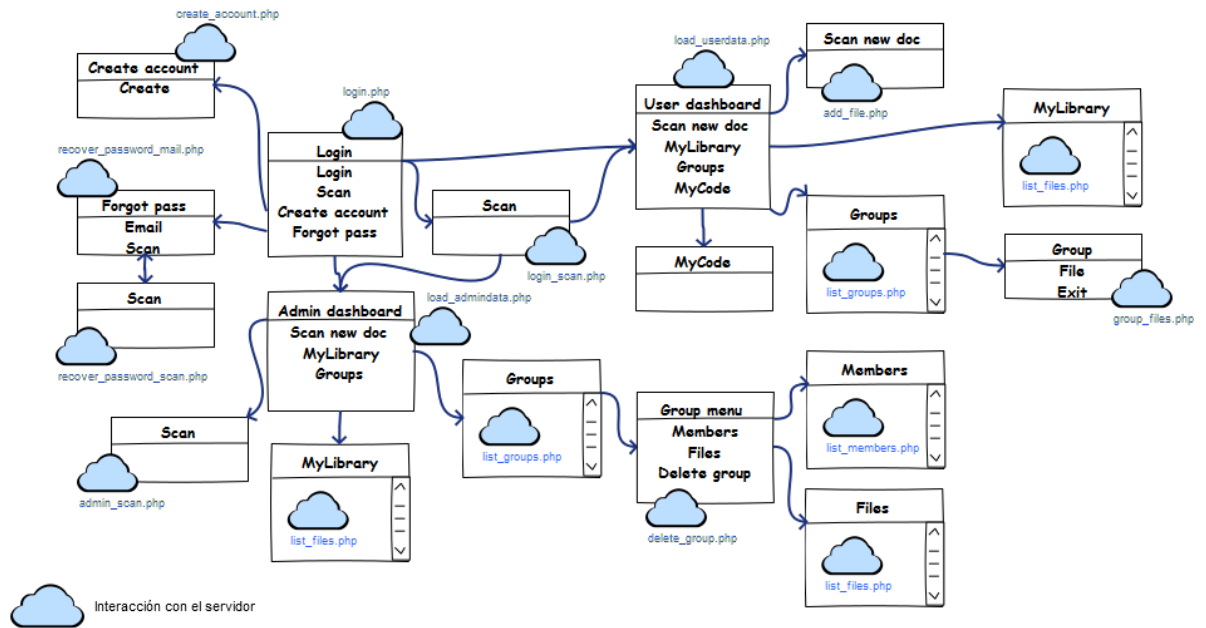


Figura 12. Mapa de navegación de la aplicación. Fuente: Propia.

Tras diseñar la navegación entre las pantallas (Figura 12), se ha invertido tiempo en recurrir a guías de diseño de aplicaciones para dispositivos móviles [16, 17], con el fin de conocer aspectos a tener en cuenta en el diseño de nuestra interfaz. A continuación se muestran los aspectos considerados:

- Se mantendrá la consistencia de elementos entre las pantallas de la aplicación. La representación gráfica de un elemento no variará a lo largo de la interacción con el sistema.
- La paleta de colores de la aplicación estará limitada a cuatro colores (a mostrar en cada pantalla como máximo).
- Para evitar confusiones, el color rojo (que normalmente se relaciona con errores) no se empleará en la aplicación.

Contando con estas premisas, se realizó el diseño de la pantalla de *login* (Figura 13) de la aplicación, que sirvió como ejemplo para el diseño y desarrollo del resto de pantallas.

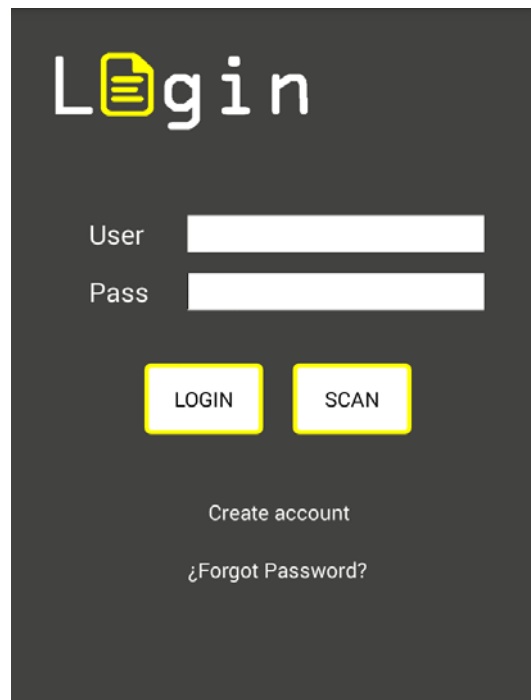


Figura 13. Diseño de la interfaz de *login*. Fuente: Propia.

3.2.4 Diseño de la lógica del sistema

Tras haber concluido el diseño del **modelo** y de la **vista**, se puede proceder al diseño del elemento que mediará entre ambas partes, el **controlador**. El controlador se encargará principalmente de:

- **Perpetuar la información** del sistema **entre** las diferentes **actividades**.
- Alojarse y **llamar a los métodos** que involucran y manejan a las clases del **modelo** (User, Admin, File y Group).
- **Construir y ejecutar** las **peticiones** que realiza la aplicación **al servidor**.
- **Recibir y parsear** las **respuestas del servidor** para **instanciar** las clases del **modelo de datos** correspondiente a dicha respuesta y permitir al sistema **trabajar con la información recibida**.

A continuación se muestra una simulación de la interacción del **controlador** con la **vista** y el **modelo** al realizar la eliminación de un documento de la librería del usuario (RF14) (Figura 14).

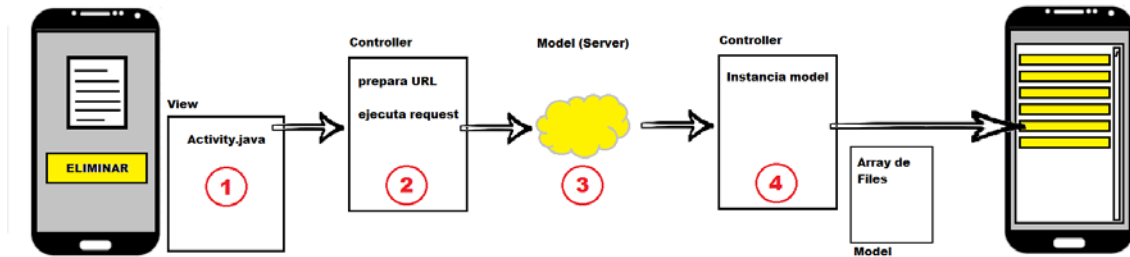


Figura 14. Simulación de interacción del controlador con el modelo y la vista. Fuente: Propia.

De modo más detallado, los procesos recogidos en la Figura 14 son:

1. La **actividad captura la pulsación** realizada sobre el botón de eliminar archivo y llama al controlador correspondiente.
2. El **controlador prepara la url** con los parámetros para enviar al servidor (en este caso la id del fichero a eliminar) y **realiza el HTTP request**.
3. El **servidor** recibe la petición, la ejecuta y **devuelve la respuesta** (en este caso un True que confirma la eliminación y un JSON con los ficheros disponibles).
4. El **controlador captura la respuesta y la interpreta**. Con los datos obtenidos se **instancia** un *Array* con los ficheros que no han sido eliminados del **servidor**. **Tras eso se inicia una nueva actividad que recibe dicho Array** y muestra actualizada la lista de ficheros disponibles.

3.3 Implementación

Tras realizar el análisis y validar el diseño del sistema, ya se cuenta con todo lo necesario para comenzar la implementación de la aplicación, a lo largo de este apartado se describirán los pasos realizados para conseguir dicho objetivo. En primer lugar, se explicará cómo se ha **preparado el entorno de desarrollo** para el proyecto a realizar. A continuación se profundizará en la parte de **codificación de las funciones del servidor** a las que llamará el controlador y que se encargarán de manejar la base de datos. Finalmente se expondrá como se ha organizado la implementación de las **clases java y ficheros XML de la aplicación Android**, proporcionando una descripción detallada de cada una de ellas así como los problemas que han podido surgir en cada parte y su resolución.

3.3.1 Preparación del entorno

Tal y como se expuso previamente en el apartado de **Análisis de herramientas y tecnologías** (3.1.3), se ha seleccionado el **Android Studio** como entorno de desarrollo del proyecto.

Una vez descargado e instalado en el ordenador de desarrollo, se puede comenzar a configurar el entorno para la implementación de la aplicación. El primer paso a realizar es la creación del proyecto, para ello se accede a la barra superior de Android Studio y se selecciona **File** para después seleccionar en el submenú la opción **New Project...** (Figura 15)

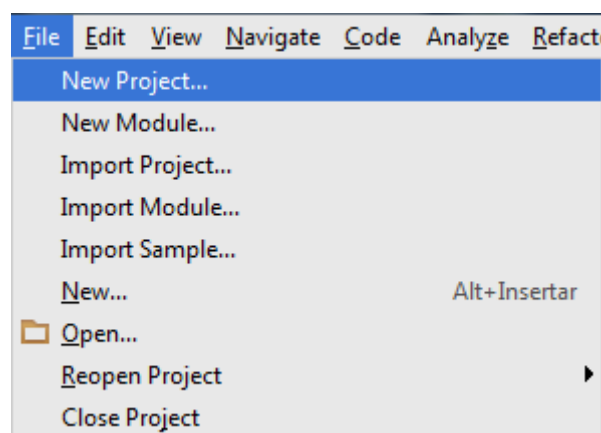


Figura 15. Preparación del entorno (I) Fuente: AndroidStudio.

Una vez en la pantalla de creación del proyecto, se seleccionan el nombre del proyecto, que en este caso se llamará **Virtual Document Storage**, el nombre del paquete del proyecto, en este caso **tfg.com**, y finalmente el directorio donde se almacenará en el ordenador (Figura 16).

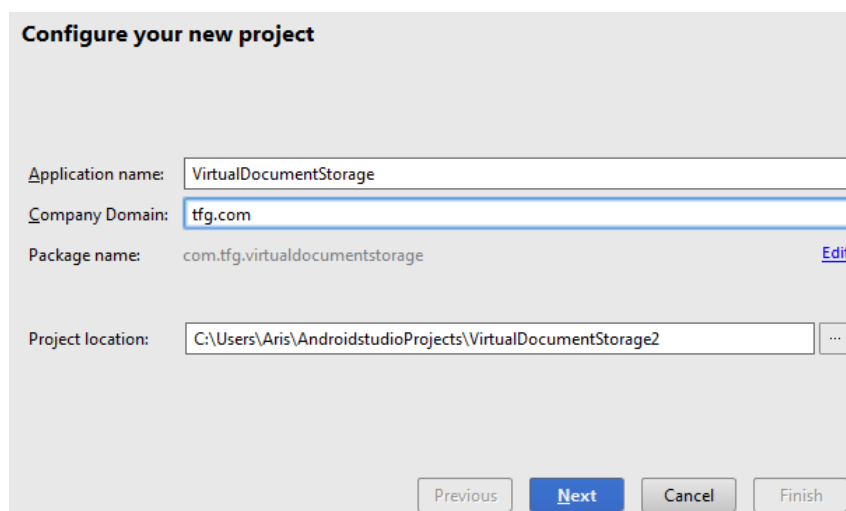


Figura 16. Preparación del entorno (II). Fuente: AndroidStudio.

Para concluir la creación del proyecto, basta con seleccionar el **mínimo SDK** objetivo de la aplicación, lo que implicará que los dispositivos Android deberán contar con un sistema operativo de versión igual o superior al seleccionado. Tal y como se adelantó en el apartado 2.1, el mínimo SDK seleccionado para nuestro sistema es el **4.0 (API 14)** (Figura 17).

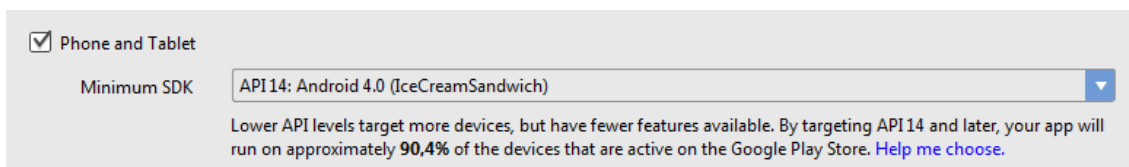


Figura 17. Preparación del entorno (III). Fuente: AndroidStudio.

Al terminar de crear el proyecto, contaremos con una serie de directorios donde se podrán incluir recursos y colocar los archivos correspondientes. No obstante, para el **patrón de arquitectura** seleccionado, se crearán **directorios personalizados** para almacenar tanto el **modelo**, como la **vista** y el **controlador**. En la siguiente imagen se muestra la estructura final del proyecto, en **rojo** los directorios que ha sido necesario crear y en **azul** los que vienen por defecto (Figura 18).

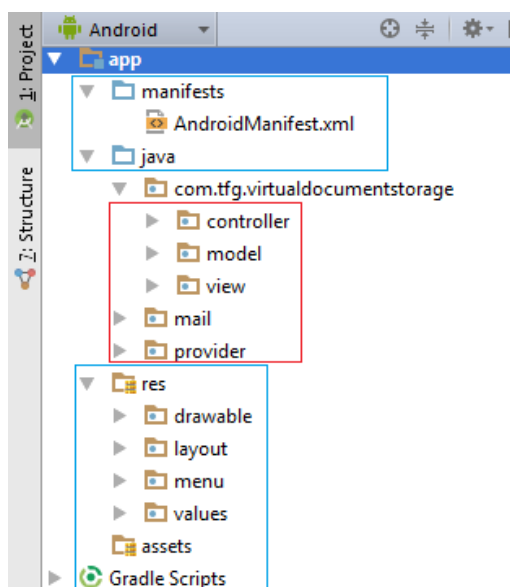


Figura 18. Preparación del entorno (IV). Fuente: AndroidStudio.

Ya se había trabajado previamente con la librería **JavaMail**, por ello se ha decidido crear los directorios **mail** y **provider** (exigidos en dicha librería) en el proyecto para futuro uso de los mismos.

Antes de comenzar la codificación, se ha decidido dejar **importadas las librerías necesarias** para el sistema que se va a implementar. Para ello se debe acceder a la estructura de ficheros del proyecto y copiar todas las bibliotecas en el directorio **libs** (Figura 19).

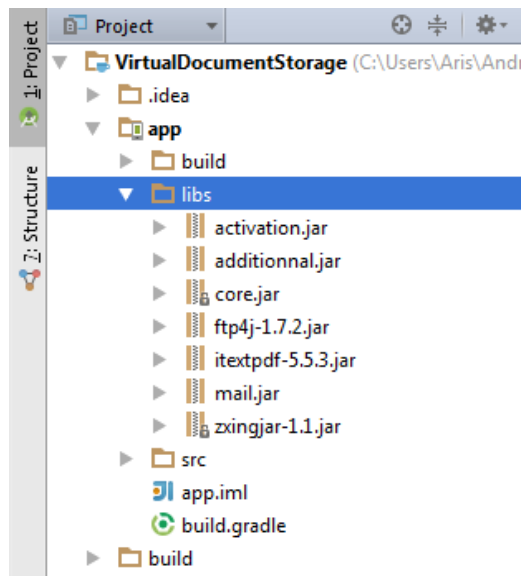


Figura 19. Preparación del entorno (V). Fuente: AndroidStudio.

Una vez preparadas las bibliotecas, se debe acceder al fichero **build.gradle** alojado en el directorio **app** y añadir la importación de las mismas, una vez incluido se habrá terminado de preparar el entorno de desarrollo:

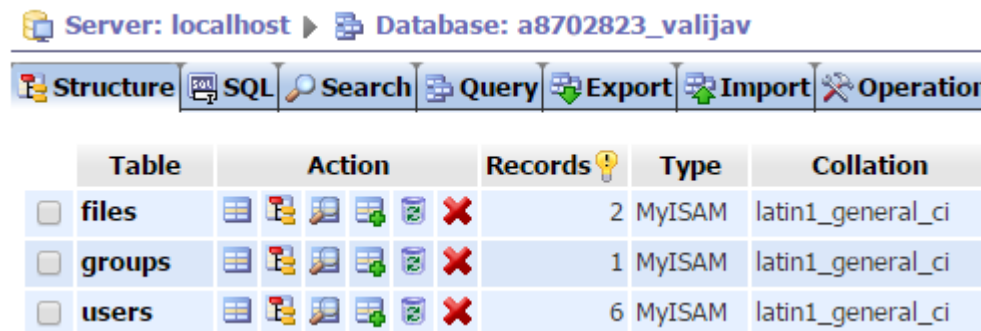
```
dependencies {
    compile 'com.android.support:support-v4:18.0.0'
    compile 'com.android.support:appcompat-v7:18.0.0'
    compile files('libs/zxingjar-1.1.jar')
    compile files('libs/core.jar')
    compile files('libs/mail.jar')
    compile files('libs/activation.jar')
    compile files('libs/additionnal.jar')
    compile files('libs/itextpdf-5.5.3.jar')
    compile files('libs/ftp4j-1.7.2.jar')
}
```

Con esto finaliza la **preparación del entorno en local**. Como se especificó en apartados anteriores nos serviremos de un **repositorio de BitBucket** donde se almacenará el código de la aplicación según avance la implementación.

3.3.2 Servidor

Antes de comenzar la codificación de la aplicación, y para facilitar la tarea de manejar la base de datos del servidor desde dicha app, se va a implementar primeramente la **funcionalidad de la API** (servidor).

Siguiendo el diseño realizado en el apartado **Diseño del modelo de datos del sistema** (3.2.2), se crea la base de datos en el servidor (a la que se ha denominado **valija_virtual**) y las tablas correspondientes (Figura 20).



The screenshot shows the phpMyAdmin interface for a server at localhost and a database named 'a8702823_valijav'. The 'Structure' tab is selected, displaying a table with the following data:

	Table	Action	Records	Type	Collation
<input type="checkbox"/>	files	[Icons]	2	MyISAM	latin1_general_ci
<input type="checkbox"/>	groups	[Icons]	1	MyISAM	latin1_general_ci
<input type="checkbox"/>	users	[Icons]	6	MyISAM	latin1_general_ci

Figura 20. Base de datos valija_virtual. Fuente: phpMyAdmin.

Una vez creada la base de datos, se puede codificar la función que se encargará de abrir la conexión con la misma mediante la creación de un **PDO** (PHP Data Object). Primeramente se definen las constantes **DB_USER**, **DB_PASSWORD** y **DB_DSN**.

```
define("PGUSER", "----");  
define("PGPASSWORD", "----");  
define("DSN", "mysql:host=mysql11.000webhost ----");
```

Tras definir las constantes, podemos implementar la **función que abra la conexión**.

```
$conexion = new PDO(DSN, PGUSER, PGPASSWORD);  
return $conexion;
```

Ésta será llamada por todas las funciones que se explicarán a lo largo de este apartado, lo que las permitirá poder realizar operaciones sobre la base de datos. Cabe destacar que **para añadir seguridad a las consultas realizadas a la base de datos, éstas serán preparadas con parámetros temporales y mediante la posterior llamada a la función bindParam() se asignarán los verdaderos valores a dichos parámetros**.

Las funciones se pueden clasificar en **tres grupos** atendiendo a la finalidad y carácter de las instrucciones que realizan en la base de datos.

3.3.2.1 Funciones de gestión

Crear cuenta: Comprueba contra la base de datos si existe el nombre de usuario introducido y en caso contrario ejecuta la consulta necesaria para registrar la nueva cuenta.

```
$ok = $db->exec("INSERT INTO users
(username,password,scan_ref,email,nombre,admin,apellido) ".
                "VALUES ('".$username."', '".
                $password."', '".
                $scan_ref."', '".
                $mail."', '".
                $name."', '".
                $admin."', '".
                $lastname."'");
```

La función se encuentra alojada en el fichero **account&login.php**

Login con username y password: Comprueba contra la base de datos si el nombre de usuario y la password introducida son correctas.

La función se encuentra alojada en el fichero **account&login.php**

Login por código QR: Comprueba contra la base de datos si el código QR escaneado corresponde con alguna cuenta registrada en la plataforma, y en caso afirmativo devuelve la información de la misma.

```
$stmt = $db->prepare("SELECT scan_ref, admin, username FROM users
"."WHERE scan_ref = :scan_ref");

$stmt->bindParam(':scan_ref', $qrcode, PDO::PARAM_STR);
$stmt->execute();

if ($stmt->rowCount()==1) {
    $linea = $stmt->fetch();
    echo $linea['admin'].$linea['username'];
}
```

La función se encuentra alojada en el fichero **account&login.php**

Comprobar disponibilidad del scan registrado: Comprueba contra la base de datos si el código QR escaneado se encuentra disponible, en caso contrario devuelve la información asociada al mismo.

La función se encuentra alojada en el fichero **qrcheck.php**

Recuperar password y username a partir del mail: Comprueba contra la base de datos la existencia del mail introducido, en cuyo caso devuelve el username y la password asociada al mismo.

La función se encuentra alojada en el fichero **qrcheck.php**

Eliminar usuario: Comprueba contra la base de datos la existencia del username introducido y de los permisos del usuario que solicita la ejecución de esta función. Si el solicitante tiene autoridad y el usuario existe, éste es eliminado.

```
$stmt = $db->prepare("DELETE FROM users  
                    ". "WHERE username = :username");  
$stmt->bindParam(':username', $username, PDO::PARAM_STR);  
$stmt->execute();
```

La función se encuentra alojada en el fichero **adminpanel.php**

Consultar usuario: Comprueba contra la base de datos la existencia del usuario introducido, en cuyo caso devuelve toda la información del mismo.

La función se encuentra alojada en el fichero **user.php**

3.3.2.2 Funciones de administración de archivos

Insertar fichero en servidor: Comprueba contra la base de datos la disponibilidad del nombre de fichero introducido, en caso afirmativo lo inserta en el servidor.

La función se encuentra alojada en el fichero **filelist.php**

Listar todos los ficheros: Devuelve todos los ficheros existentes en el servidor.

```
$stmt = $db->prepare("SELECT * FROM files ");  
$stmt->execute();
```

La función se encuentra alojada en el fichero **filelist.php**

Listar todos los ficheros de un usuario: Devuelve todos los ficheros existentes en el servidor cuyo autor sea el introducido. Simplemente añade una condición a la consulta anterior.

```
$autor = $_REQUEST["autor"];
$stmt = $db->prepare("SELECT * FROM files ".
                    "WHERE autor = :autor");
$stmt->bindParam(':autor', $autor, PDO::PARAM_STR);
```

La función se encuentra alojada en el fichero **filelist.php**

Eliminar fichero: Comprueba contra la base de datos la existencia del nombre de fichero y de autor introducido, en caso de que exista el fichero y coincida el autor dicho fichero se eliminará.

```
$stmt = $db->prepare("SELECT file_name FROM files ".
                    "WHERE file_name = :file_name
                    AND autor = :autor");
$stmt->bindParam(':file_name', $file_name, PDO::PARAM_STR);
$stmt->bindParam(':autor', $autor, PDO::PARAM_STR);
$stmt->execute();
```

Si existe:

```
$stmt = $db->prepare("DELETE FROM files ".
                    "WHERE file_name = :file_name");
$stmt->bindParam(':file_name', $file_name, PDO::PARAM_STR);
$stmt->execute();
```

La función se encuentra alojada en el fichero **filepanel.php**

Modificar permisos: Comprueba contra la base de datos la existencia del fichero introducido, en cuyo caso actualiza los permisos de acceso del mismo.

```
$stmt = $db->prepare("UPDATE files ".
                    "SET access = '$permisos' ".
                    "WHERE file_name = :file_name");
$stmt->bindParam(':file_name', $file_name, PDO::PARAM_STR);
$stmt->execute();
```

La función se encuentra alojada en el fichero **filepanel.php**

Modificar tags: Comprueba contra la base de datos la existencia del fichero introducido, en cuyo caso actualiza las etiquetas de clasificación del mismo.

```
$stmt = $db->prepare("UPDATE files ".  
                    "SET keywords = '$tags'".  
                    "WHERE file_name = :file_name");  
$stmt->bindParam(':file_name', $file_name, PDO::PARAM_STR);  
$stmt->execute();
```

La función se encuentra alojada en el fichero **filepanel.php**

3.3.2.3 Funciones de administración de grupos

Ver grupos a los que pertenece un usuario: Comprueba contra la base de datos la existencia del usuario introducido y devuelve los grupos a los que pertenece.

```
$stmt = $db->prepare("SELECT id, admin FROM groups ".  
                    "WHERE user = :username");  
$stmt->bindParam(':username', $username, PDO::PARAM_STR);  
$stmt->execute();
```

La función se encuentra alojada en el fichero **group.php**

Eliminar grupo: Comprueba contra la base de datos la existencia de un grupo y lo elimina de la base de datos.

```
$stmt = $db->prepare("DELETE FROM groups ".  
                    "WHERE id = :groupname");  
$stmt->bindParam(':groupname', $groupname, PDO::PARAM_STR);  
$stmt->execute();
```

La función se encuentra alojada en el fichero **group.php**

Añadir al grupo: Comprueba contra la base de datos la existencia del grupo, en cuyo caso añade al usuario al mismo. En caso contrario el grupo será creado y el usuario será añadido de igual manera.

La función se encuentra alojada en el fichero **group.php**

Listar usuarios del grupo: Comprueba contra la base de datos la existencia del grupo, en caso afirmativo lista a todos los usuarios que son miembros del mismo.

La función se encuentra alojada en el fichero **group.php**

Eliminar usuario del grupo: Comprueba contra la base de datos la existencia del grupo y si el usuario introducido pertenece al mismo, en cuyo caso será eliminado de éste

```
$stmt = $db->prepare("DELETE FROM groups ".  
                    "WHERE id = :groupname ".  
                    "AND user = :username");  
$stmt->bindParam(':username', $username, PDO::PARAM_STR);  
$stmt->bindParam(':groupname', $groupname, PDO::PARAM_STR);  
$stmt->execute();
```

La función se encuentra alojada en el fichero **group.php**

Devolver el administrador de un grupo: Comprueba contra la base de datos la existencia del grupo y en caso de que exista devuelve el nombre del administrador del mismo.

La función se encuentra alojada en el fichero **group.php**

Disponibilidad de un grupo: Comprueba contra la base de datos la disponibilidad del nombre de grupo introducido y devuelve el resultado.

La función se encuentra alojada en el fichero **group.php**

3.3.3 MVC

Antes de comenzar el desarrollo de la API, dejamos listo el entorno sobre el que desarrollar nuestro sistema, ahora se debe continuar con la parte de la implementación más importante, **la aplicación**.

Se ha seguido un riguroso proceso de **implementación dividido en varios pasos** bien delimitados que han otorgado **orden a la codificación** de las diferentes funcionalidades. Dichos pasos se explican a continuación.

3.3.3.1 Implementar el modelo de la aplicación

El primer paso a realizar es la codificación del Modelo de datos, tarea sencilla siguiendo el diseño realizado en el apartado 3.2.2.1. La única característica a destacar, es que todas las clases del modelo **implementan** la clase **Serializable**, esto permitirá guardar clases completas en caché o añadir las clases como **información extra** de una actividad a otra.

```

public class Group implements Serializable {

    private String group_name;
    private ArrayList<User> users;
    private String admin;

    public Group(String group_name, String admin) {
        this.group_name = group_name;
        this.users = new ArrayList<User>();
        this.admin = admin;
    }
}

```

3.3.3.2 Actualizar el fichero de manifiesto

Tras implementar las clases Java del modelo de datos, lo siguiente que se ha de realizar es la **declaración de los componentes** en el **fichero de manifiesto** de la aplicación. Este fichero se encarga de dar permisos a la aplicación, declara la versión actual de la misma y establece la versión mínima de Android que debe tener el dispositivo para ejecutarla. Además recoge las declaraciones de actividades del sistema, lo que permitirá compilarlas y poder ejecutarlas.

Con la creación del proyecto en el apartado **Preparación del entorno** (3.3.1), se autogeneró un fichero de manifiesto con información básica, como el nombre y código de la versión y el SDK mínimo y SDK objetivo de la aplicación. Para el funcionamiento de la aplicación, se deben añadir los permisos pertinentes de uso que necesitaran las funcionalidades de ésta.

```

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_INTERNAL_STORAGE" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.FLASHLIGHT" />

```

A medida que avance la codificación será necesario crear nuevas **Actividades**, que **serán declaradas en el manifiesto** dentro de la estructura **<application>**. En la declaración se definirán atributos como el nombre de la actividad o la etiqueta que aparecerá en el ActionBar durante su ejecución.


```
<activity
    android:name=".view.CreateAccount"
    android:label="@string/app_name"
    android:screenOrientation="portrait"
    android:configChanges="orientation|keyboardHidden">
</activity>
```

Para la aplicación a implementar, se ha decidido permitir únicamente la orientación de pantalla vertical, por lo que se ha añadido el atributo **screenOrientation**. Además para forzar el inicio de la actividad con el teclado oculto se ha añadido **keyboardHidden** a **configChanges**.

Siguiendo el ejemplo de declaración de una actividad expuesto, se procederá a **declarar todas las actividades** de la aplicación **a medida que se vayan creando**. La única actividad que contará con una **declaración especial** será la de **inicio de la aplicación** a la que se añadirá el siguiente código.

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

3.3.3.3 Implementar la vista de la actividad

La implementación de la vista de la actividad se dividirá en dos fases, la **implementación de la interfaz gráfica en XML** y la **implementación de la clase JAVA que contendrá dicha interfaz** y que servirá de nexo para poder interactuar con la misma.

Para la implementación de la interfaz gráfica se empleará la herramienta de **renderizado en tiempo real** incluida en el **Android Studio** (Figura 21).

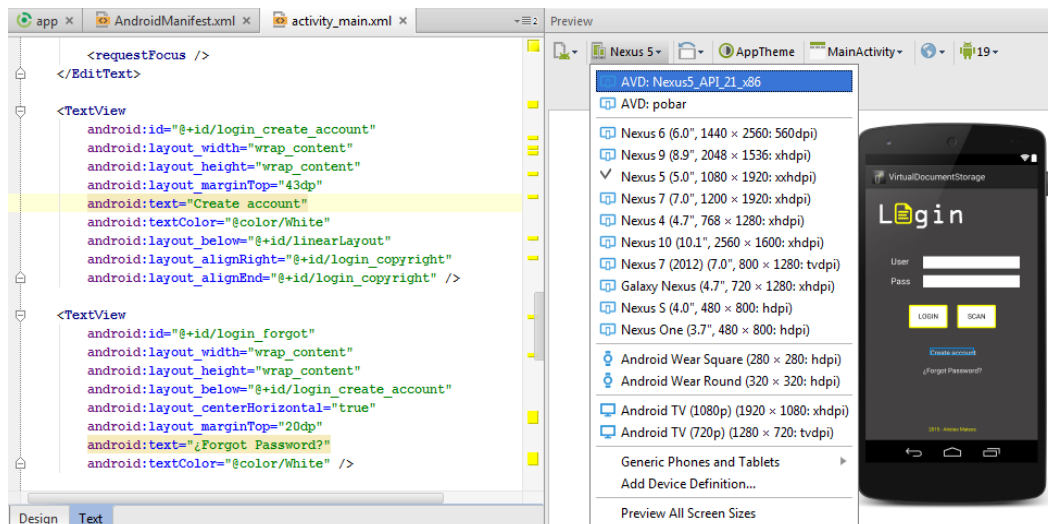


Figura 21. Herramienta de renderizado para la implementación de la GUI. Fuente: AndroidStudio.

En el XML se codificarán todos los componentes de la interfaz implementados a partir de objetos de las clases **View** y **ViewGroup**, asignando IDs significativas para mayor claridad de código. En todo momento la implementación estará orientada a cumplir con el **RNF03**, permitiendo a los usuarios **ejecutar la aplicación** sin problemas **en cualquier dispositivo** independientemente de la resolución o tamaño de la pantalla. Tras realizar la codificación del XML, se empleará la opción **Pre visualizar todos los tamaños de pantalla** (Figura 22) para comprobar la portabilidad de la implementación y poder ajustarla si es necesario.



Figura 22. Pre visualización de tamaños de pantalla para portabilidad. Fuente: AndroidStudio.

Una vez codificada la interfaz gráfica, se implementará la clase **JAVA de la actividad** que la contendrá y que **escuchará la interacción** con los elementos de la misma para **llamar posteriormente a las funciones correspondientes del controlador**.

Cada una de las actividades se implementa en una clase Java que hereda de la clase base **Activity** de Android. Para establecer como contenido de la Actividad la interfaz implementada anteriormente, se hace uso de la función **setContentView()** cuyo argumento es el ID de recurso, asignado al XML.

```
setContentView(R.layout.activity_main);
```

Para escuchar las interacciones del usuario con la **GUI**, la actividad implementará la interfaz **View.OnClickListener** y mediante la sobre escritura del método **onClick()**, se codificarán las llamadas a las funciones correspondientes al objeto con el que se ha interactuado.

```
/*En onCreate() se establece el Listener en el objeto de la GUI*/

Button button = (Button) findViewById(R.id.login_button);
button.setOnClickListener(this);

/*Y en onClick() se codifican las reacciones de la aplicación ante la
interacción previa*/

public void onClick(View v) {
    if(v == findViewById(R.id.login_button)){...}
}
```

Se ha empleado la sobre escritura del método **onStart()**, para la implementación del **RF30** (Actualización de listas en tiempo real) en las actividades que lo requerían. El uso de este método difiere del convencional **onCreate()** pues al volver a una actividad desde otra que acaba de finalizar, el código incluido en **onStart()** se ejecutará de nuevo, al contrario que el situado en **onCreate()**. Es por ello que el **código que actualizará las listas** de ficheros, usuarios y grupos **se localizará dentro del método onStart()**.

Para **inicializar nuevas actividades** desde la actividad actual, se han empleado dos métodos, **startActivity()** y **startActivityForResult()**. El **primero** ha permitido iniciar actividades cuyo **resultado de ejecución** no afecta o **no es relevante** para la ejecución de las funcionalidades de la aplicación, mientras que el **segundo** devuelve un **resultado** que **capturará la actividad** que lo ha llamado y que reaccionará dependiendo del valor del resultado mediante la **implementación del método onActivityResult()**. A ambos métodos se les puede añadir información extra que compartirán la actividad invocadora y la invocada, mediante la función **putExtra()**.

3.3.3.4 Implementar el controlador de la actividad

Tras finalizar la implementación de la Actividad, con sus escuchadores y código asociados a los mismos, lo único que queda es implementar las funciones que actuarán de nexo entre dicha actividad y el modelo de nuestro sistema.

La **operación con las funciones del modelo**, codificadas en el apartado 3.3.3.1 son referenciadas desde métodos del controlador, y por tanto **llamadas directamente desde el controlador instanciado en la actividad**.

Son **las operaciones con la base de datos del servidor** las que exigen un estudio más detallado. Los **accesos** al servidor se realizarán mediante **request HTTP**, donde se añadirán **parámetros a una URL** de forma dinámica cargando dichos parámetros en un **Vector de clave/valor**. Tras definir la URL se realiza la **ejecución del cliente HTTP** y se **obtiene la respuesta** de la petición al servidor. Esta respuesta recibida será **interpretada por otra función del controlador**, que devolverá el resultado tras *parsear* dicha respuesta.

Cabe destacar que **las llamadas** a las funciones del controlador que interactúan con el **servidor**, **se invocarán desde una subclase Asíncrona** que extienda de la clase **AsyncTask**, y que será definida en la actividad desde donde se va a realizar la llamada. De esta forma **se evitará que la aplicación se bloquee** al iniciar la interacción con el servidor en el hilo principal.

3.3.3.5 Codificaciones a destacar

File Transfrer Protocol: El servicio de *hosting* elegido para el proyecto nos ofrece un alto número de prestaciones de forma gratuita, a cambio de sacrificar cierta seguridad. El servidor no da soporte SFTP por lo que la transferencia de archivos se realizará siguiendo un protocolo FTP normal.

Para la implementación del protocolo por parte de la aplicación, se hace uso de la librería **ftp4j-1.7.2** especificada en apartados anteriores. Una vez importada la librería, se incluirá la declaración de las constantes **FTP_HOST**, **FTP_USER** y **FTP_PASS**, y la función **uploadFile()** en la actividad donde se realice la generación del documento a transferir.

```
public String uploadFile(File fileName){  
  
    StrictMode.ThreadPolicy policy = new  
        StrictMode.ThreadPolicy.Builder().permitAll().build();  
  
    StrictMode.setThreadPolicy(policy);  
  
    FTPClient client = new FTPClient();  
  
    try {
```

```

        client.connect(FTP_HOST, 21);
        client.login(FTP_USER, FTP_PASS);
        client.setType(FTPClient.TYPE_BINARY);
        client.changeDirectory("/public_html/UploadFiles");

        client.upload(fileName, new MyTransferListener());

        return URL + fileName.getName();
    }
    catch (Exception e) {
        e.printStackTrace();
        try {
            client.disconnect(true);
        }
        catch (Exception e2) {
            e2.printStackTrace();
        }
    }
    return null;
}

```

Code Scan: Con el fin de implementar la funcionalidad de escaneo de códigos personalizados, se hace uso de la librería **zxingjar-1.1** especificada en apartados anteriores. Para su uso, se debe invocar a la actividad proporcionada por la librería importada, desde la actividad donde se quiera recibir el código escaneado.

```

IntentIntegrator.initiateScan(MainActivity.this, R.layout.capture,
                                R.id.viewfinder_view, R.id.preview_view, true);

```

Posteriormente, se recibirá el resultado de la invocación, en la actividad invocadora.

```

protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    String scancode="";
    switch (requestCode) {
        case IntentIntegrator.REQUEST_CODE:
            IntentResult scanResult =
                IntentIntegrator.parseActivityResult(requestCode,
                                                        resultCode, data);

            if (scanResult == null) {
                return;
            }
            final String result = scanResult.getContents();
            scancode = result;
            break;
        default:
    }
}

```

3.3.4 Incrementos de software utilizable

Tal y como se especificó en el apartado de **Planificación del proyecto** (1.3), se ha seguido un **marco de desarrollo ágil** de software con una estrategia de **desarrollo incremental**. Los incrementos se produjeron en **periodos de dos semanas** al final de los cuales se entregó al usuario una aplicación que cumplía los requisitos especificados para dicho incremento.

A continuación se determina la **funcionalidad que se cumplió en cada incremento** realizado:

- **Sprint 1:** Funcionalidad de **registro en el sistema**, *login* por scan o por username y password y **funcionalidad de recuperación de contraseña** por mail o por scan.
- **Sprint 2:** Funcionalidad de **Usuario**, escaneado de documentos, acceso a librería y acceso a código personal.
- **Sprint 3:** Funcionalidad de **Administrador**, escaneado de documentos de terceros y **creación de grupos**.
- **Sprint 4:** **Funcionalidad completa de la aplicación**, incluyendo administración de grupos y de usuarios.

4. Evaluación del sistema

En este capítulo se detallará el **plan de pruebas** al que se ha sometido a la aplicación **durante y tras su desarrollo**, con el fin de **detectar errores en el sistema** y realizar una evaluación real de uso donde **evidenciar discordancias con los requisitos** especificados en el apartado de **Análisis** (3.1). Posteriormente se detallarán los resultados de las pruebas realizadas por los usuarios al testear los **incrementos de software utilizable** resultantes de la finalización de cada **Sprint**.

4.1 Plan de pruebas

El objetivo de este apartado es analizar los tipos de pruebas realizadas sobre la aplicación, explicando el procedimiento de cada una y aportando ejemplos de las mismas. Este plan de pruebas se ha seguido en cada Sprint realizado.

4.1.1 Pruebas unitarias

Las pruebas unitarias tienen como objetivo la comprobación de la lógica, la funcionalidad y la correcta especificación de cada módulo implementado. Para llevar a cabo estas pruebas se codificó un **módulo extra** encargado de **llamar a las funciones del resto de módulos** implementados y de **evaluar la lógica interna** de dichas funciones (caja blanca).

PU01: Correcta creación del objeto Usuario con los datos introducidos a través de la vista correspondiente y creación de una URL válida.

La función recibe correctamente los parámetros y construye el objeto sin complicaciones. Además construye la URL con dichos parámetros y llama a la función del controlador correspondiente. **Prueba superada.**

PU02: Correcta interpretación de la respuesta del servidor, que devuelve los campos de file.

La función recibe correctamente la respuesta del servidor y la interpreta eliminando los caracteres que sea necesario y creando el objeto file que rellenará posteriormente con los datos parseados. **Prueba superada.**

4.1.2 Pruebas de integración

Las pruebas de integración tienen como **objetivo testear la agrupación de los diferentes módulos de la aplicación** y comprobar el flujo entre las interfaces. Para llevar a cabo estas pruebas se emplearan pruebas de caja negra, donde se **controlarán las entradas** de las mismas y **analizarán las salidas** producidas.

PI01: Función que recibe el usuario y la password y llama al módulo que realiza la petición de login al servidor.

Entradas: User (go) y Password (go)

Salida esperada: True

La función interpreta correctamente la entrada y llama a la función de llamada al servidor. Esta función realiza la llamada y devuelve la respuesta.

Salida obtenida: True. **Prueba superada.**

PI02: Función que recibe el usuario y llama al módulo que realiza la petición de checkUser (comprueba la existencia del usuario) al servidor.

Entradas: User (tester)

Salida esperada: -1

La función interpreta correctamente la entrada y llama a la función de llamada al servidor. Esta función realiza la llamada y devuelve la respuesta.

Salida obtenida: -1. **Prueba superada.**

4.1.3 Pruebas de sistema

Las pruebas de sistema tienen como objetivo **testear la integración hardware y software del sistema**. Se ha instalado, ejecutado e interactuado con la aplicación en diferentes dispositivos. La correcta implementación de la interfaz gráfica ha permitido usar la aplicación en todos los dispositivos testeados sin problemas.

PS01: Xiaomi Redmi Note 4G (Prueba superada)

- Versión de Android 4.4.2
- Pantalla de 5.5 pulgadas
- Resolución de 720x1280

PS02: Acer Iconia Tab A1-810 (Prueba superada)

- Versión de Android 4.2
- Pantalla de 7.9 pulgadas
- Resolución de 1024x768

PS03: Lenovo K3 Note(K50-t5) (Prueba superada)

- Versión de Android 5.0
- Pantalla de 5.5 pulgadas
- Resolución de 1920x1080

4.1.4 Pruebas de validación

Las pruebas de validación tienen como objetivo comprobar la **concordancia de los requisitos** especificados de la aplicación, con la **funcionalidad real implementada**. Las pruebas se realizaron comprobando uno a uno los requisitos funcionales y no funcionales especificados en el apartado 3.1.2 de este documento. Las pruebas de validación fueron realizadas **sobre la aplicación resultante de cada sprint**, antes de proporcionársela al usuario para que éste la testease.

4.2 Prueba de los incrementos de software utilizables

Las pruebas sobre las aplicaciones de funcionalidad parcial desarrolladas en cada incremento serán probadas por usuarios reales, realizando de esta forma las **pruebas de aceptación**. Estas pruebas nos indicarán en cada Sprint si el producto implementado se ajusta a lo que el usuario ha solicitado.

4.2.1 Incremento 1

PA01

Usuario: Dámaso Plaza Fernández

Experiencia del usuario con aplicaciones móviles: Media

Funcionalidad a probar:

- Creación de cuenta en el sistema
- Logueo por scan
- Logueo por username y password
- Recuperación de contraseña por mail

- Recuperación de contraseña por scan

Problemas encontrados:

- Usuario no entiende por qué se solicita el mail
- Usuario no sabe si ha pulsado el botón de login

Conclusiones:

- Se debe cambiar la animación al pulsar los botones para que se evidencie la pulsación
- Se debe añadir un dialogo de *Login In...* mientras la aplicación interactúa con el servidor

4.2.2 Incremento 2

PA02

Usuario: Dámaso Plaza Fernández

Experiencia del usuario con aplicaciones móviles: Media

Funcionalidad a probar:

- Escaneado de documentos
- Acceso a librería del usuario
- Acceso a código personal del usuario

Problemas encontrados:

- La conexión a internet del usuario hace que el tiempo de descarga de los documentos escaneados sea excesiva.

Conclusiones:

- Como ya se especificó en el RNF05, la conexión a internet del usuario es independiente de la aplicación y no se hará cargo de los retrasos ocasionados por la misma.

4.2.3 Incremento 3

PA03

Usuario: Dámaso Plaza Fernández

Experiencia del usuario con aplicaciones móviles: Media

Funcionalidad a probar:

- Funcionalidad de Administrador
- Escaneado de documentos de terceros como administrador
- Creación de grupos

Problemas encontrados:

- Ningún problema a destacar

4.2.4 Incremento 4 (Aplicación final)

PA04

Usuario: Dámaso Plaza Fernández

Experiencia del usuario con aplicaciones móviles: Media

Funcionalidad a probar:

- Funcionalidad completa de la aplicación

Problemas encontrados:

- Ningún problema a destacar

Conclusiones:

- La **implementación del proyecto ha sido un éxito** y el usuario está satisfecho con el resultado.

5. Conclusiones

En este apartado se expone la **valoración personal** del proyecto en función tanto del **proceso de desarrollo** del mismo, como del **resultado final** obtenido. Seguidamente se detallarán los **conocimientos y valores** que me ha **aportado**, a nivel personal, la realización del sistema. Finalmente se listarán **mejoras a introducir** en futuras versiones.

5.1 Valoración personal del proyecto

Se ha desarrollado un **sistema completo**, que cumple los requisitos especificados y que **satisface** las subjetivas **necesidades de un usuario** que busca la sencilla interacción con una aplicación. Atendiendo a las pruebas realizadas y a mi opinión tras vivir el desarrollo del proyecto, considero que éste ha sido **finalizado de forma satisfactoria** con una **valoración personal muy positiva**.

5.2 Conocimientos y valores aportados

El proyecto desarrollado destaca entre otros que haya realizado debido a su uso de diferentes tecnologías con las que no había trabajado previamente, como el **escáner de códigos QR**. El proceso de desarrollo se caracterizó por el estudio en profundidad de todas las tecnologías a emplear como el escáner mencionado, la **biblioteca ftp para java** o el **cliente de correo Android**. Dicho estudio me sirvió para asimilar estas tecnologías y emplearlas con soltura, cabe destacar que el conocimiento adquirido del **escáner de códigos personalizados** ya he podido **usarlo en un ámbito profesional**.

El hecho de desarrollar un proyecto que cuenta con parte de desarrollo web, me ha permitido profundizar mi conocimiento en el **backend** y me ha hecho ser consciente del valor de los mismos, teniendo que **aprender a adaptarme a las limitaciones** de seguridad, rendimiento o espacio, impuestas por el plan gratuito contratado.

Finalmente el trabajar en esta herramienta ha afianzado mi interés por el desarrollo de aplicaciones Android y me ha **motivado para el inicio de una trayectoria profesional** en dicho campo.

5.3 Futuras versiones

Las limitaciones económicas influyeron notablemente en el desarrollo de la aplicación, principalmente en la seguridad. Para futuras versiones se estudiará **el pago de una tarifa** que nos permita disponer de un **servidor más seguro**, en el que introducir **protocolos** como el **SFTP y SSL**. Cabe destacar que la falta de acoplamiento entre los componentes funcionales de la aplicación permiten incorporar este tipo de mejoras fácilmente.

En relación a la app, en futuras versiones se trabajará en una **clasificación automática de documentos** mediante el uso de un **OCR**, que reconozca los caracteres del documento y permita analizarlos y buscar patrones en base a los cuales realizar un etiquetado automático de la información. Asimismo, y tal y como hemos destacado a lo largo del texto, la aplicación permite la inclusión de otros sistemas de almacenamiento en la nube. Esto es de gran relevancia en el contexto tecnológico actual en el que se tiene a combinar sistemas *cloud* privados y públicos, así como otros medios de almacenamiento gestionados todos ellos de modo centralizado.

Bibliografía

- [1] “Android: A visual history.” (2011). Retrieved May 8, 2015, from <http://www.theverge.com/2011/12/7/2585779/android-history>
- [2] “T-Mobile G1.” (2008). Retrieved May 11, 2015, from <http://www.xataka.com/moviles/tmobile-g1>
- [3] “Scrum methodology.” (2009). Retrieved May 12, 2015, from <http://scrummethodology.com/>
- [4] “Mobile operating system.” (n.d.). Retrieved May 13, 2015, from http://en.wikipedia.org/wiki/Mobile_operating_system
- [5] Guru. (2015). “Top 10 Mobile Phones Operating Systems.” Retrieved May 13, 2015, from <http://www.shoutmeloud.com/top-mobile-os-overview.html>
- [6] ALPOMA. (2006). “El Pantelógrafo.” Retrieved from <http://www.alpoma.net/tecob/?p=338>
- [7] Henry, A. (2015). “Mobile Document Scanning Apps.” Retrieved May 13, 2015, from <http://lifehacker.com/five-best-mobile-document-scanning-apps-1691417781>
- [8] Strickland, J. (2008). “How Cloud Storage Works.” Retrieved May 14, 2015, from <http://computer.howstuffworks.com/cloud-computing/cloud-storage.htm>
- [9] “Cloud storage.” (n.d.). Retrieved May 14, 2015, from http://en.wikipedia.org/wiki/Cloud_storage
- [10] “File sharing.” (n.d.). Retrieved May 14, 2015, from http://en.wikipedia.org/wiki/File_sharing
- [11] “Bokodes, otra alternativa a los códigos de barras.” (2009). Retrieved May 14, 2015, from <http://www.consumer.es/web/es/tecnologia/internet/2009/08/25/187124.php>

- [12] "Different Types of Barcodes." (n.d.). Retrieved May 14, 2015, from <http://www.makebarcode.com/specs/speclist.html>
- [13] "Usuario (Informática)." (n.d.). Retrieved May 19, 2015, from http://www.ecured.cu/index.php/Usuario_%28Inform%C3%A1tica%29
- [14] "Model View Controller explained." (2009). Retrieved May 18, 2015, from <http://www.tomdalling.com/blog/software-design/model-view-controller-explained/>
- [15] "MVC Design Pattern." (n.d.). Retrieved May 20, 2015, from <https://teamtreehouse.com/library/build-a-blog-reader-android-app/exploring-the-masterdetail-template/the-modelviewcontroller-mvc-design-pattern-2>
- [16] Krug, S. (n.d.). *"Don't Make Me Think. A Common Sense Approach to Web Usability."*
- [17] Tidwell, J. (n.d.). *"Designing Interfaces: Patterns for Effective Interaction Design."*
- [18] "Nube híbrida." (n.d.). Retrieved from <https://routingtheworld.wordpress.com/spanish-content/nube-publica-nube-privada-nube-hibrida-y-nube-super-hibrida/>
- [19] "Nearly one in 10 children gets first mobile phone by age five, says study" (2013). Retrieved from <http://www.theguardian.com/money/2013/aug/23/children-first-mobile-age-five/>

